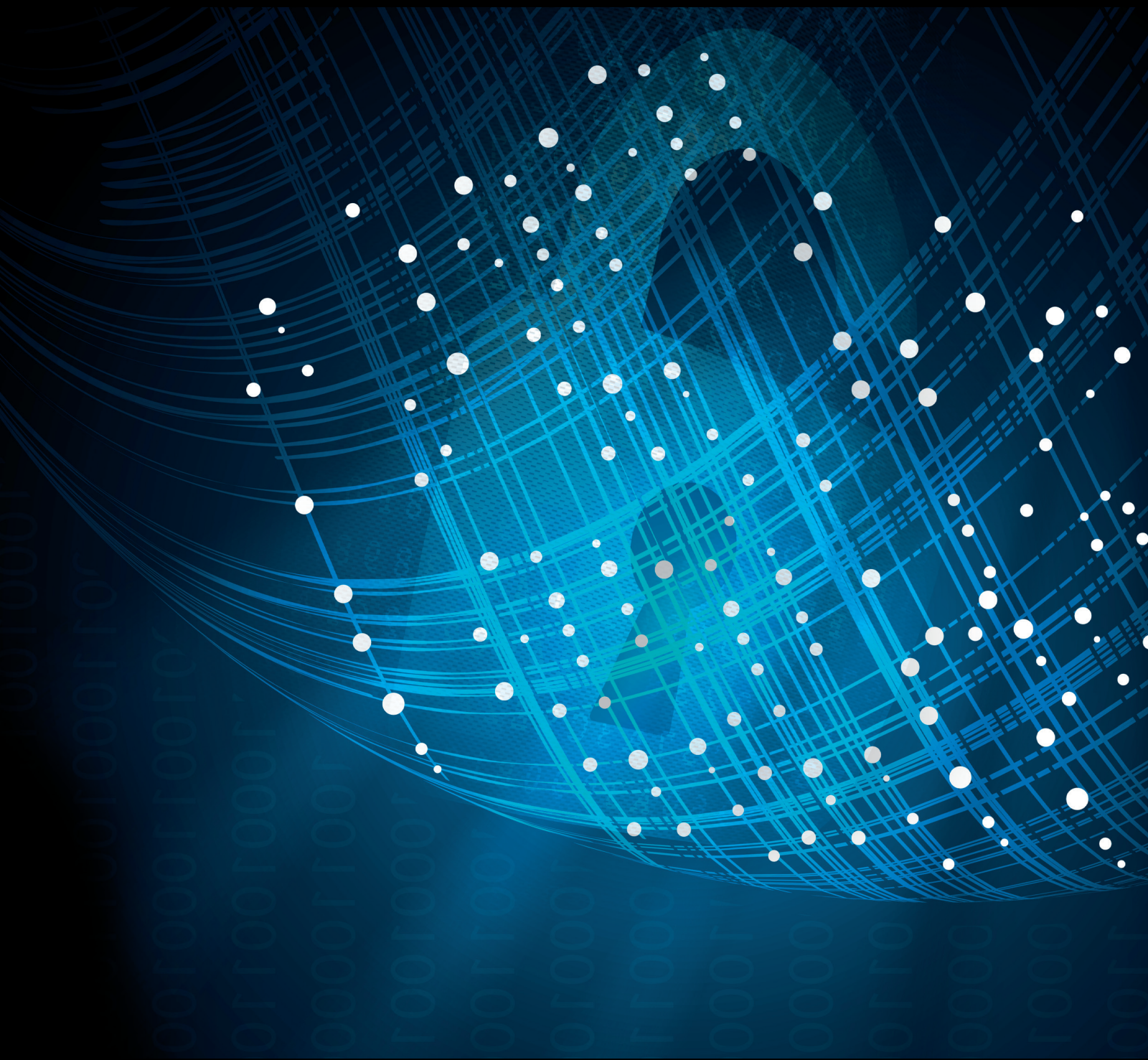
Security and Communication Networks

Data-Driven Cybersecurity

Lead Guest Editor: Hyoungshick Kim
Guest Editors: Jungwoo Ryoo, Sebastian Schrittwieser, and Surya Nepal



Data-Driven Cybersecurity

Security and Communication Networks

Data-Driven Cybersecurity

Lead Guest Editor: Hyoungshick Kim
Guest Editors: Jungwoo Ryoo, Sebastian Schrittwieser, and Surya Nepal

Copyright © 2019 Hindawi Limited. All rights reserved.

This is a special issue published in “Security and Communication Networks.” All articles are open access articles distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Chief Editor

Roberto Di Pietro, Saudi Arabia

Associate Editors

Jiankun Hu, Australia
Emanuele Maiorana, Italy
David Megias, Spain
Zheng Yan, China

Academic Editors

Saed Saleh Al Rabaee, United Arab Emirates
Shadab Alam, Saudi Arabia
Goutham Reddy Alavalapati, USA
Jehad Ali, Republic of Korea
Jehad Ali, Saint Vincent and the Grenadines
Benjamin Aziz, United Kingdom
Taimur Bakhshi, United Kingdom
Spiridon Bakiras, Qatar
Musa Balta, Turkey
Jin Wook Byun, Republic of Korea
Bruno Carpentieri, Italy
Luigi Catuogno, Italy
Ricardo Chaves, Portugal
Chien-Ming Chen, China
Tom Chen, United Kingdom
Stelvio Cimato, Italy
Vincenzo Conti, Italy
Luigi Coppolino, Italy
Salvatore D'Antonio, Italy
Juhriyansyah Dalle, Indonesia
Alfredo De Santis, Italy
Angel M. Del Rey, Spain
Roberto Di Pietro, France
Wenxiu Ding, China
Nicola Dragoni, Denmark
Wei Feng, China
Carmen Fernandez-Gago, Spain
AnMin Fu, China
Clemente Galdi, Italy
Dimitrios Geneiatakis, Italy
Muhammad A. Gondal, Oman
Francesco Gringoli, Italy
Biao Han, China
Jinguang Han, China
Khizar Hayat, Oman
Azeem Irshad, Pakistan
M.A. Jabbar, India
Minho Jo, Republic of Korea
Arijit Karati, Taiwan
ASM Kayes, Australia
Farrukh Aslam Khan, Saudi Arabia
Fazlullah Khan, Pakistan
Kiseon Kim, Republic of Korea
Mehmet Zeki Konyar, Turkey
Sanjeev Kumar, USA
Hyun Kwon, Republic of Korea
Maryline Laurent, France
Jegatha Deborah Lazarus, India
Huaizhi Li, USA
Jiguo Li, China
Xueqin Liang , Finland
Zhe Liu, Canada
Guangchi Liu, USA
Flavio Lombardi, Italy
Yang Lu, China
Vincente Martin, Spain
Weizhi Meng, Denmark
Andrea Michienzi, Italy
Laura Mongioi, Italy
Raul Monroy, Mexico
Naghmeh Moradpoor, United Kingdom
Leonardo Mostarda, Italy
Mohamed Nassar, Lebanon
Qiang Ni, United Kingdom
Mahmood Niazi, Saudi Arabia
Vincent O. Nyangaresi, Kenya
Lu Ou, China
Hyun-A Park, Republic of Korea
A. Peinado, Spain
Gerardo Pelosi, Italy
Gregorio Martinez Perez, Spain
Pedro Peris-Lopez, Spain
Carla Ràfols, Germany
Francesco Regazzoni, Switzerland
Abdalhossein Rezai, Iran
Helena Rifà-Pous, Spain
Arun Kumar Sangaiah, India
Nadeem Sarwar, Pakistan
Neetesh Saxena, United Kingdom
Savio Sciancalepore, The Netherlands

De Rosal Ignatius Moses Setiadi, Indonesia
Wenbo Shi, China
Ghanshyam Singh, South Africa
Vasco Soares, Portugal
Salvatore Sorce, Italy
Abdulhamit Subasi, Saudi Arabia
Zhiyuan Tan, United Kingdom
Keke Tang, China
Je Sen Teh, Australia
Bohui Wang, China
Guojun Wang, China
Jinwei Wang, China
Qichun Wang, China
Hu Xiong, China
Chang Xu, China
Xuehu Yan, China
Anjia Yang, China
Jiachen Yang, China
Yu Yao, China
Yinghui Ye, China
Kuo-Hui Yeh, Taiwan
Yong Yu, China
Xiaohui Yuan, USA
Sherali Zeadally, USA
Leo Y. Zhang, Australia
Tao Zhang, China
Youwen Zhu, China
Zhengyu Zhu, China

Contents

Hindawi
Security and Communication Networks
Volume 2019, Article ID 1901548, 21 pages
https://doi.org/10.1155/2019/1901548

WILEY | Hindawi

Research Article

CBR-Based Decision Support Methodology for Cybercrime Investigation: Focused on the Data-Driven Website Defacement Analysis

Mee Lan Han, Byung Il Kwak, and Huy Kang Kim

Graduate School of Information Security, Korea University, Seoul, Republic of Korea

Correspondence should be addressed to Huy Kang Kim; cenda@korea.ac.kr

Received 25 March 2019; Revised 21 August 2019; Accepted 2 December 2019; Published 20 December 2019

Guest Editor: Jungwoo Ryoo

Copyright © 2019 Mee Lan Han et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Criminal profiling is a useful technique to identify the most plausible suspects based on the evidence discovered at the crime scene. Similar to offline criminal profiling, in-depth profiling for cybercrime investigation is useful in analysing cyberattacks and for speculating on the identities of the criminals. Every cybercrime committed by the same hacker or hacking group has unique traits such as attack purpose, attack methods, and target. These unique traits are revealed in the evidence of cybercrime; in some cases, these unique traits are well hidden in the evidence such that it cannot be easily perceived. Therefore, a complete analysis of several factors concerning cybercrime can provide an investigator with concrete evidence to attribute the attacks and narrow down the scope of the criminal data and grasp the criminals in the end. We herein propose a decision support methodology based on the case-based reasoning (CBR) for cybercrime investigation. This study focuses on the massive data-driven analysis of website defacement. Our primary aim in this study is to demonstrate the practicality of the proposed methodology as a proof of concept. The assessment of website defacement was performed through the similarity measure and the clustering processing in the reasoning engine based on the CBR. Our results show that the proposed methodology that focuses on the investigation enables a better understanding and interpretation of website defacement and assists in inferring the hacker's behavioural traits from the available evidence concerning website defacement. The results of the case studies demonstrate that our proposed methodology is beneficial for understanding the behaviour and motivation of the hacker and that our proposed data-driven analytic methodology can be utilized as a decision support system for cybercrime investigation.

1. Introduction

Advanced persistent threat (APT) attacks, stealthily and continuously controlled by hackers or hacking groups targeting a specific entity, remain as a challenging threat, particularly to the companies or organizations that handle sensitive funding and information. When successful, APT attacks can have a catastrophic impact on critical infrastructures, such as banking, broadcasting system, and mass media sites. This impact is not speculative or theoretical—in fact, it is supported by various real-world incidents and actual attacks. For instance, in February 2016, a group of hackers stole $81 million from the Central Bank of Bangladesh through its account at the Federal Reserve Bank of New York through an APT attack which targeted constantly the SWIFT payment system for a year [1]. Furthermore, in May 2017, the WannaCry ransomware, another type of APT attack, spread due to the vulnerability in the Microsoft Server Message Block (SMB; the message format used to share folders and files and so on in Microsoft Windows OS). This attack caused catastrophic consequences, such a standstill and disruption of online work in hospitals, companies, and several government agencies. According to Symantec's 2017 annual report [2], the SWIFT case and the WannaCry ransomware case were perhaps launched by the Lazarus group that could be affiliated to the DarkSeoul (DS) case in 2013 and the Sony Pictures Entertainment (SPE) case in 2014. Symantec found that the hacking skills in the SWIFT case were very similar to those used by the Lazarus group, presumably one of the North

Korea's state-sponsored hacking group; the report also found that the malware of the WannaCry ransomware case was related to the one used by the Lazarus group [3]. In the Operation Blockbuster report released by Novetta in 2016, the Lazarus group was reported to hypothetically come in two basic classes—the features known as the wipers and the DDoS malware [4]. The noticeable features of these attacks underpin our interest in the Lazarus group's attack related to the DS case in 2013 and the SPE case in 2014.

On March 20, 2013, in the DS case, the DS's attack destroyed approximately 48,700 computerized and networked equipment items, such as PCs, servers, and network devices of major banks and TV broadcasters in South Korea. South Korea suffered a coordinated strike by a simple but very effective and destructive malware called Wiper A, B, and C [5]. In certain Windows OS environments, the wiper scripts attempted to remove any directories after attempting to overwrite each file with a specific string pattern (i.e., "HASTATI," "PRINCIPES," or "PR!NCPES") [6, 7]. In another incident initiated by the Lazarus group, the SPE was hacked by the self-named Guardians of Peace (#GOP) hacker group. Several malware analysis groups reported that the GOP attack was also related to the North Korean cyber army [8]. The malware used in this attack contained strings written using the Romanization of Korean words (i.e., Korean words were spelled using Latin letters following the English pronunciation). Of note, while the Korean language as spoken in North Korea and South Korea is linguistically identical, there are several important differences in terms of vowels and consonants, phonetic notation, and word spacing [9]. In the aforementioned case, the Romanized words captured in the malware were having various contemporary North Korean words.

From 2009 to 2017, along with the attacks mentioned above, the Lazarus group launched many other attacks (see Figure 1 for further details).

There have been numerous attempts in industry and academia to do hacker profiling and to handle attack incidents. These approaches can be categorized into the following three types: the human-centric analysis, malware-centric analysis, and case-centric analysis. The human-centric analysis approach focuses on hacker network analysis. Known hacker activities (e.g., message postings and discussion) on the hacker communities provide a clue to identify key actors by their reputation. In addition, it can classify the tendency of a hacker based on social networking methods [10, 11]. Unlike the human-centric analysis, the malware-centric approach primarily assumes that the same malware and its variants could be developed by the same or closely similar hacker groups. Among others, features such as API call sequence and control flow can be used to estimate the similarity between the newly detected malware and the known malware [12–14]. In fact, many previous studies on hacker profiling have primarily focused on using information derived from the analysis of the malware itself. While malware analysis could provide information about a malware's functionality and its similarity with the previously known malware family, tracing and analysing hacker information based on the malware centric could have the limitation where the core information can be circumvented. The last approach is the case-centric analysis, and our methodology falls into this category. Overall, only several proposals can be applied to the traditional investigation method, such as criminal profiling methods, to the cyber incident investigation; however, many systematic approaches are currently under development. From the viewpoint of the cyber intelligence analysis, the case-centric analysis has the advantage of making it possible to understand the purpose of attack campaigns; it is important to build profiles of attackers as with other methods of analysis. When performed successfully, such characterizations can facilitate estimating and predicting the attacker's next target in advance.

Based on this insight, the present study proposes the CBR-based decision support methodology for cybercrime investigation. In terms of data, website defacement attack cases occurring between 1998 and 2015 were retrieved from the public archival site zone-h.org (an archive of defaced websites, http://www.zone-h.org). After crawling web resources of the Hypertext Markup Language (HTML) type, data preprocessing for data parsing and data cleaning were performed to amend incomplete, improperly formatted, or duplicate data records. The case vector was designed to intuitively express defaced website cases collected from the public archival site. The reasoning engine will be able to start the major work only after completing the data preprocessing and the case vector design. The similarity measurement based on CBR was performed in them. The clustering algorithm was performed to group-abstracted crime cases into classes of similar cases. Based on the results concerning the DS and SPE cases, we evaluated the performance of the framework for cybercrime investigation by measuring the similarity and clustering algorithm. The results demonstrated that the proposed methodology can be used as a Decision Support System (DSS) to obtain meaningful information about the most similar past cases and related hacker groups.

The main contributions of the present study are summarized as follows:

(i) We present a CBR-based decision support methodology for cybercrime investigation. With the proposed cybercrime investigation scheme, security analysts can find past attack cases that are most similar to a given attack and thus obtain insights to uncover the networks of cybercrime related to website defacement. To deliver high-value intelligence, we adopt the data-driven analytic system.

(ii) We demonstrate the clustering processing and visualization. The clustering processing enables an investigator to efficiently explore large data and interpret the results. Furthermore, the visualization helps an investigator to intuitively recognize crime patterns.

(iii) We propose that it is possible to measure the similarity score and to perform the clustering algorithm by transforming unstructured data (i.e., web defacement cases) into calculable structured data.

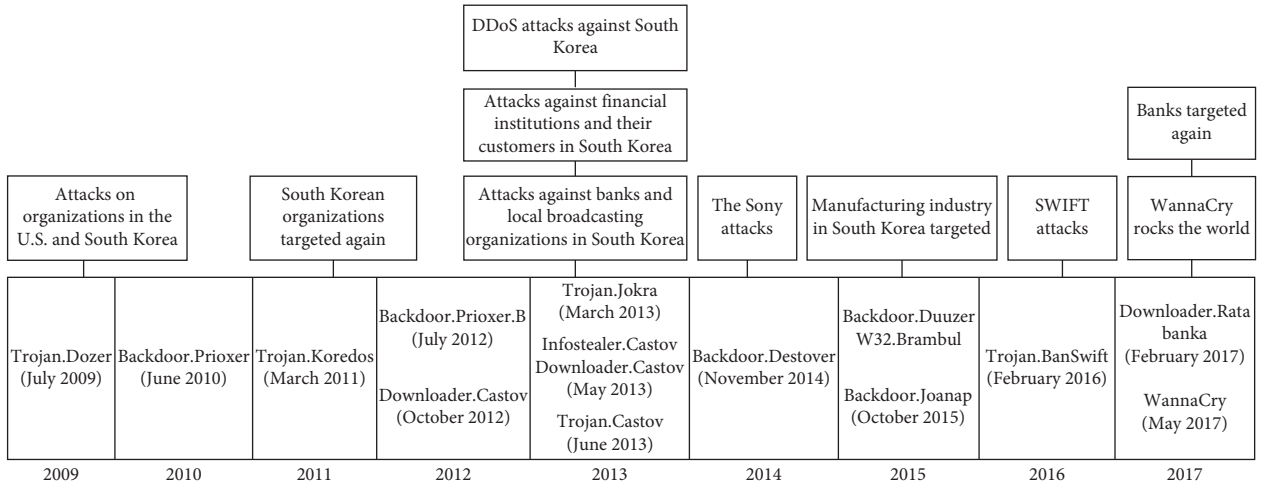


Figure 1: Timeline of the Lazarus group activities from 2009 to 2017.

(iv) We report case studies based on the real dataset gathered from the zone-h.org site to demonstrate the various aspects of our proposed algorithm. Finally, to foster further research, our dataset (the dataset for cybercrime investigation: focused on the data-driven website defacement analysis, http://ocslab.hksecurity.net/Datasets/web-hacking-profiling) is made publicly available [15, 16].

The rest of the paper is organized as follows. Section 2 provides a summary of the literature related to our work. The detailed methodology is described in Section 3. Section 4 reports the experimental results and analysis based on the case study. The limitations of our work and a discussion on the proposed approach are presented in Section 5. Finally, Section 6 concludes the paper and suggests directions of further research.

2. Related Work

In this section, we primarily highlight the previous studies closely related to the CBR and review two streams of literature on traditional criminal profiling and cybercrime profiling. We also elaborate the data mining-based cybercrime profiling pertaining to the following: (1) the CBR studies that help better understand our research context; (2) traditional criminal profiling and cybercrime profiling review that allow us to obtain an elusive criminal or a concealed clue; (3) data mining-based cybercrime profiling literature that can support and theoretically reinforce our methodology.

2.1. Case-Based Reasoning. CBR is a method that uses past experiences or cases to solve new problems. Even when the new problems are not exactly identical to the previous cases, CBR can suggest a partial solution to the new problems [17]. CBR can be categorized as a data-mining technique, as it can classify the given samples and predict the result for a new case. As case studies are intuitive and easily understood by humans, CBR has long been used in many fields, including customer technical support, medical case search, and legal case search. The general model of the four-step CBR process [18] is shown in Figure 2.

The four phases are as follows:

(i) Retrieve: given a new website defacement case, relevant cases are retrieved from the knowledge base to solve the case at hand

(ii) Reuse: solutions from previous website defacement cases are mapped for reuse

(iii) Revise: on mapping and testing previous solutions to the target case, the solutions are revised to consider the changes in the cases

(iv) Retain: after the solution has been successfully adapted to the problem, a meaningful experience is stored as a new case in the knowledge base

CBR starts with a given set of cases for training, forms generalizations of the given examples, and subsequently identifies the commonalities between the retrieved case and the target case. When applied to the website defacement case composed of descriptive and nominal data, it can effectively determine the commonality from the crawled hacking cases and quickly search the nearest related case. Furthermore, CBR can be used to search the most similar cases and retrieve past solutions from the latest response cases. CBR facilitates security administrators to make better decisions. For example, Kim et al. proposed the DSS for an incident response based on CBR [19].

CBR has been extensively used in several areas, such as management for product development, medicine, and in engineering applications [20–22]. In addition, several CBR approaches were available for cyber incidents profiling. For instance, Kim et al. proposed an intelligent system that can measure the similarity between the past and new attacks. In their work, the author(s) demonstrated such capability in uncovering zero-day attacks using the string similarity analysis of the captured packet-level data [23]. Horsman et al. proposed the CBR-FT framework which is a method for collecting and reusing past digital forensic investigation information to highlight likely evidential areas on a suspect

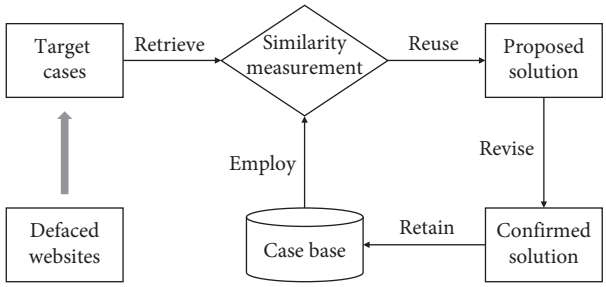


Figure 2: CBR process used in cybercrime investigation, employing knowledge base that is reused over similar new cases and retained for later use.

operating system. It enables an investigator to help quickly and precisely decide where to search for evidence [24].

2.2. Traditional Criminal Profiling and Cybercrime Profiling. Profiling is used in various sectors of the society to investigate a criminal's mentality. Criminal profiling is a profiling technique for criminal investigation based on the psychological and behavioural patterns of a criminal [25, 26]. The criminal aspects and crime factors can be identified through the evidences and insights of the psychological and behavioural bias [27]. In the field of criminology, the widely used profiling technique is called the Modus Operandi (MO). It is used to describe a suspect's behaviour and evidence elements in crime. That is, it means how a suspect commits their crimes. The Modus Operandi changes based on the offender's criminal conduct and interaction with the surrounding such as time, date, and location of crime. Moreover, it evolves based on how the offender reaches his/her victim [28, 29].

Based on only the traditional criminal profiling techniques and empirical knowledge, it is difficult for a cybercrime investigator to reduce the error of the investigative process and to untangle the complexity of a cybercrime. However, if the investigator is provided with sufficient information and detailed analysis data to understand the unclear motivation and the elusive pattern related to the cybercrime, they can infer the reason(s) of the crime at stake and produce both general and specific outlines of the criminal [26]. The cybercrime network and characteristics can be important indicators to differentiate between key figures in the cybercrime organizations and those of passing interest. In addition, their activity periods and message content patterns of the participants in an illegal community can support the investigator to carefully identify and scrutinize the key figure in the cybercrime network [30, 31]. By automating cybercrime profiling and data-mining methods of analysis through a cross-analysis of various behavioural patterns, we can anticipate potential criminal activities and identify new profiles that pose serious threats to the community. Furthermore, data-mining methods such as entity extraction, clustering/classification technique, and social network analysis make it possible to efficiently explore large data. Network visualization enables an investigator to intuitively recognize the crime pattern [32–34].

In general, the accuracy of CBR depends on the quality of the collected data, and the overall accuracy is difficult to evaluate [35]. Although the effectiveness of data-driven investigation can decrease owing to the dynamic and fast-evolving crime patterns, understanding the hidden correlations and latent behaviour in such data using large data analytic techniques is another promising direction in research. Accordingly, many law enforcement agencies have been adopting future crime prediction systems based on the statistics about weather, cleanliness, location, demographic distribution, education level, and wealth-level information. Based on the crime prevention through environmental design (CPTED) theory [36], many pieces of data correlated with the crime are collected and analysed to estimate the crime probability. However, while many data-driven approaches to support traditional criminal profiling are available, only several research efforts have focused on cybercrime profiling.

2.3. Data-Driven Cybercrime Profiling. In addition to the traditional criminal profiling for offline crime investigations, various profiling techniques have been developed; in these techniques, it is assumed that cybercriminals also show similar behavioural and psychological characteristics. Owing to the recent advances in data-mining and machine-learning algorithms, many studies regarding criminal pattern detection, classification, and clustering have emerged. The methods used in these studies include, among others, entity extraction, clustering, association rule mining, deviation detection, and classification of social network analysis. A combination of the traditional method and a newer method enables the pattern identification from both structured and unstructured data. For instance, entity extraction is used to understand concealed patterns in the data such as texts, images, and audio data. Furthermore, clustering is used to group objects into classes with similar characteristics [37, 38]. In addition, unsupervised methods, such as the self-organizing map (SOM), are used to support the results of the traditional criminal profiling [39]. In cases where the criminal and the related cases are known, supervised learning is applied [40]. However, although many advances have occurred in big data analytics and machine learning, these approaches are limited in supporting real-time processing, as they require high computing power to handle a large volume of training data. In fact, the large volume of crime data is a considerable challenge for the investigator in terms of gaining the appropriate understanding of a complicated relationship or in terms of a timely response. However, despite the limitations of this approach, data mining yields valid, useful, and appropriate results. By data preprocessing such as data cleaning, data integration, and data transformation, it intends to reduce noisy data, as well as incomplete and inconsistent data. It helps to uncover and conceptualize the concealed or latent crime patterns. By improving the efficiency of crime data understanding and reducing errors in the results afforded by the data-mining method, the investigator can perform reasoning, timely judgment, and quick problem solving [41].

CBR is also used to provide the reasoning power to search similar previous cases [25, 42]. However, biased or imperfect collected data deteriorate the quality of the decision support provided by CBR. Therefore, in many cases, setting the weight of the selected features is based on empirical knowledge, which can be subsequently used to enable the detection and analysis of crime patterns from the temporal crime activity data. Using clustering and classification techniques, as well as speculative models for searching similar crime cases in the past, investigators can easily extract useful information from the unstructured textual dataset [43]. Hence, investigators must collect and continuously update the comprehensive crime data.

Clustering is the task of determining a similar group in the data. Clustering includes supervised learning types. Zulfadhilah et al. compared four types of clustering algorithms: K-means, hierarchical clustering, SOM, and Expectation Maximization algorithm (EM clustering)—based on their performances. They concluded that the K-means algorithm and the EM algorithm are better than the hierarchical clustering algorithm. In general, partitioning algorithms such as the K-means and EM algorithm are highly recommended for use in large-size data [44]. In summary, the clustering algorithm can facilitate the investigator in detecting crimes patterns and accelerate crime solving. The weighting scheme for attributes can handle the limitations of the clustering techniques [45].

3. Methodology

In this section, we present the detailed scheme of decision support methodology for cybercrime investigation with the focus on the website defacement cases. A conceptual framework and its process are illustrated in Figure 3. The scheme is proceeded by the following three steps: data preprocessing, case vector design, and reasoning engine. First, we provide a brief outline of the dataset and describe the merits of the website defacement data. Also, we summarize the preprocessing for data parsing and cleaning regarding the collected data type. Next, we designed the case vector and chose the significant features to apply the reasoning performance. Finally, the reasoning engine has various functionalities, and it is intended for the grouping (clustering) of cases based on their similarity.

3.1. Preprocessing. As part of the proposed analytical framework, we have developed a crawler to automatically collect 212,093 website defacement cases from the zone-h.org site. Many website defacement cases are being daily recorded in the archive page of the zone-h.org site. Each case registered in the archive page provides information (i.e., IP address, Domain, Date, OS, Notifier, and Web server) of the same format through each mirror page. First of all, the crawler collects all public information relevant to each case. Thereafter, on accessing the domain site, it saves data in the raw format of the HTML source. After crawling the web resources of raw data, the data preprocessing is performed to amend incomplete, improperly formatted, or duplicate data records. More specifically, there are various tag attributes in the HTML source. Encoding and Font data are extracted through the <charset> and <font-style> tag of the HTML elements set between <head> and </head> tag in the HTML source. Also, image, sound file, and the linked site are extracted through the <font-family>, <img>, and <href> tag of them set between <body> and </body> tag in the HTML source. The web resources as original raw data were parsed and cleaned depending on the relevant case vector (see Figure 4). After cleaning the data, some significant data fields were selectively stored in the system's case database.

The selected data fields were related to the information about the website defacement date, related IP address, target domain, target system OS, and web server version; these aspects have proven to be useful for cyberattack investigations [46]. Specifically, the encoding method and the font whom the HTML source contains were necessary to speculate on the attacker's regional information. For example, if messages remaining in a defaced website are written in ISO/IEC 8859 encoding, we can subsequently infer that the hackers' language is German, Spanish, or Swedish. Furthermore, depending on whether all the messages are written in the same encoding method, the used special characters such as β or $\tilde{n}$ or $\mathring{a}$ can be used as a clue for guessing attacker's origin. In general, encodings from Windows-1250 to Windows-1258 are used in the central European languages, as well as in Turkish, Baltic languages, and Vietnamese. By contrast, GB encoding is used in Chinese, HKSCS encoding is used in Taiwanese, and EUC-KR or ISO-2022-KR encoding is used in Korean [47]. In addition to the font and encoding information, the text, image, audio, and video found in the messages are also necessary parameters for the case identification.

3.2. Case Vector Design. We designed the case vector in two types concerning the similarity measure and clustering processing. The case vector is summarized in Table 1. The features of various aspects such as the font, web server, thanks-to, notifier (hackers or hacking groups), as well as the features such as the encoding, IP address, domain, attack date, and OS, were extractable from the public archival site zone-h.org. Generally, more diverse features can be a significant factor for investigating relationships and associations among hackers or hacking groups and the scale and the density/intensity of the hacker community. However, such a premise has some shortcomings. The importance or the weight of all features may be different depending on the criterion. Also, if all features are important, machine-learning algorithms such as clustering or classification are difficult to perform in reality because of the high computational cost for analysing. Despite having similar meanings, some of the features can be reperformed unnecessarily. To this end, the dimensionality reduction and the feature selection were performed in the present study paper. After a thorough review by security experts, the significant features were selected for the case vector of website defacement cases. The detailed explanation of the dimensionality reduction and the feature selection is as follows.

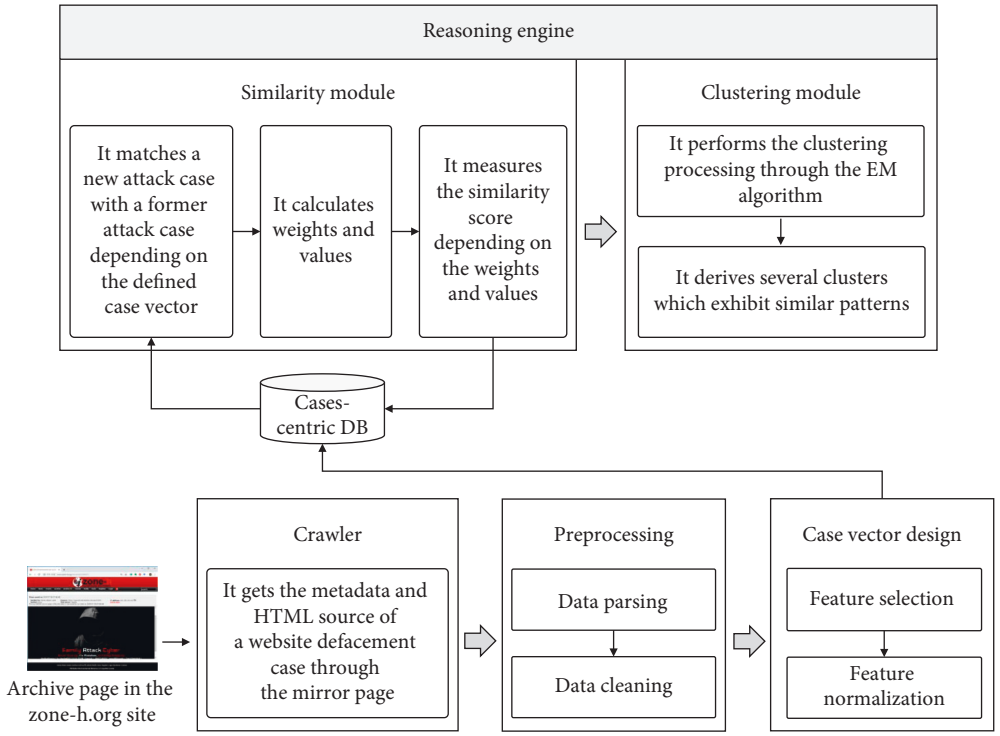


Figure 3: Proposed analytical framework for the data-driven website defacement cases.

Full View | Item View | Script Output |

	Num	Date	Domain	gTLD	ccTLD	IP_Octet1	IP_Octet2	IP_Octet3	IP_Octet4	OS	Encoding	Notifier
1	1	37364.93354	gocomp	com		209	197	254	55	linux		Mr.X
2	2	37267.52771	outfitters	com						linux	utf-8	null
3	3	41558.821	0.static.wix	com		216	139	213	107	linux		HueHueHue
4	4	39820.82729	1.soodoo	com		59	36	101	41	window		Xiaoyu
5	5	41871.3169	132		ir	5	61	24	52	linux		Adr-Elite404
6	6	39981.98212	001bolanhui	net	cn	58	215	85	94	window	WestEurope	federal-atack.org
7	7	37116.06927	001hq	com						unix	utf-8	Kebracho
8	8	39981.98212	001zhanxiaohui	com		58	215	85	94	window	WestEurope	federal-atack.org
9	9	41562.53906	2.huangshi	gov	cn	221	233	125	180	linux		Iranian_Dark_Coders_Team
10	10	41562.53903	4.huangshi	gov	cn	221	233	125	180	linux		Iranian_Dark_Coders_Team
11	11	41562.53892	5.huangshi	gov	cn	221	233	125	180	linux		Iranian_Dark_Coders_Team
12	12	40166.94075	7.hypnotic		hu	217	116	47	129	linux		K-E-H
13	13	41562.47785	1.taranefa1	com		69	43	161	134	linux		Hidden Pain
14	14	39991.80419	0101tech	com		98	130	83	69	linux	WestEurope	TechnicaL
15	15	39993.10155	010fsw	com		61	156	39	67	window		federal-atack.org
16	16	39982.84795	115318878	com		216	46	178	77	linux	Arabic	TheWayEnd

Figure 4: Sorted dataset through the preprocessing.

In the Windows operating system, if a specific font is not designated as the tag inside the HTML code, such as the <font-family> property, the characters on a website page may appear as broken. In particular, some of the fonts among the Chinese characters' cultural area depend on the character encoding (e.g., font-family: Gulim, MingLiU, and STHeiti) [48]. Similar to the encoding feature, although this characteristic may be the key evidence to uncover a correlation between the victim and the attacker, it is extremely rare in each of the collected website defacement cases. Therefore, it is not suitable as a case vector for cybercrime investigation. Meanwhile, in the case of a web server, it provides HTML, CSS, JavaScript, etc. when a client requests a web page using the web server. While the Apache and IIS web servers are primarily used in the Windows environment, the LiteSpeed web server is primarily used in the Linux environment and the Enterprise web server is primarily used in the UNIX environment. Therefore, the web server is selectively dependent on the OS environment. As with the font feature described before, since the web server feature could not be found in the collected website defacement cases, it was not suitable as a case vector for cybercrime investigation. Finally, although the case vector concerning thanks-to and notifier can be used to analyse a hidden network between the hackers and hacker groups, the analysis of a network among hackers and hacking groups through them should be addressed in future research.

As a result, we defined the case vector by dividing into two types, i.e., a version for the similarity measure and a version for the clustering processing. As the features of the case vector, the encoding, IP address, domain (i.e., service name, gTLD, and ccTLD), attack date, and OS were used in the similarity measure. However, the encoding, gTLD, ccTLD, and OS were used in the clustering processing. The encoding is a case vector that provides decisive clues related to the attacker's region information. In the case of the IP

Table 1: Case vector design, highlighting two groups of features.

Case vector	Used in process		Description
	S	C	
Encoding	O	O	It is used to represent the different types of language information on the computer. It determines the usable characters and the methods to express them. The feature was normalized based on MS Windows and the ISO character set
IP address	O	N/A	A unique number that allows devices on the network to identify and communicate with each other
Domain			
Service name	O	N/A	The service name is individually made with a different name depending on the service categories such as gTLD or ccTLD
gTLD	O	O	The gTLD feature was normalized depending on the element having the same meaning (e.g., .go, .gob, and .gobr feature were normalized into .gov)
ccTLD	O	O	The ccTLD is a unique code assigned to the domain name that represents the country, specific region, or an international organization The ccTLD normalized by the continent is used in the clustering process, and the original ccTLD is used in the similarity process
Date	O	N/A	The attack date performed by the hacker or the hacking group
OS	O	O	A part of a computer system that manages all hardware and software (e.g., Windows, Linux, and UNIX)

S, similarity measure; C, clustering processing.

address and domain, it gives clues related to the victim's location and position. Furthermore, the attack date gives clues to the relation between the attacker and the victim. The detailed explanation of key features is provided in Table 1.

The normalization result of various feature elements stored in the raw form of the HTML source is presented in Figure 5. In the case of encoding, ISO series and MS Windows series are applied by normalizing depending on the encoding used in each region or country. In the case of gTLD, it was applied by normalizing depending on the groups or organizations with similar characteristics. In the case of ccTLD, it was applied by normalizing depending on each continent. Although the compression and normalization of features enable making the analysis, such as clustering processing and similarity measure, simple and clear, on the contrary, it may also bring about the loss of information in the original data or make it more difficult to analyse in detail.

3.3. Reasoning Engine. In the reasoning process, the reasoning engine first performs a similarity search based on CBR. Discrete similarity scores are defined to calculate the distance of nominal data (e.g., IP address and domain). Algorithm 1 shows how the similarity module operates by comparing a retrieved website defacement case and all cases in the cases-centric DB on a case-by-case basis. Subsequently, the reasoning engine evaluates the similarity score between the given new attack case vector and vectors of other attack cases. Next, the reasoning engine performs clustering to group-abstracted crime cases into classes of similar crime cases. In crime investigation, a cluster grouped as similar crime case subsets helps to infer crime patterns and speeds up the process of solving a crime due to a better understanding of a complicated relationship or in terms of a timely response. In the present study, we implemented the reasoning engine consisting of two processing entities: the similarity measure processing and the clustering algorithm processing (see below for further details).

3.3.1. Similarity Measure. As the similarity measure based on the CBR algorithm, we proposed the similarity algorithm operated by comparing a retrieved website defacement case and all cases in the cases-centric DB. To begin with, if one of the retrieved cases (RC: a new case) is given and there are "n" cases in the cases-centric DB (TCs: all cases in the cases-centric DB), a comparison between RC and TCs are conducted as "n" times. We defined the extent of similarity between RC and TCs as a numeral value from "0" to "1," where "0" means that RC and TC are unrelated and "1" means that RC and TC are identical. Similarity score $(0 < S < 1)$ specifies the extent of similarity between RC and TC. If the similarity score is much closer to "1," RC and TC are more analogous to each other. In the event of multiple case vectors, similarity can be expressed as a weighted sum of case vectors:

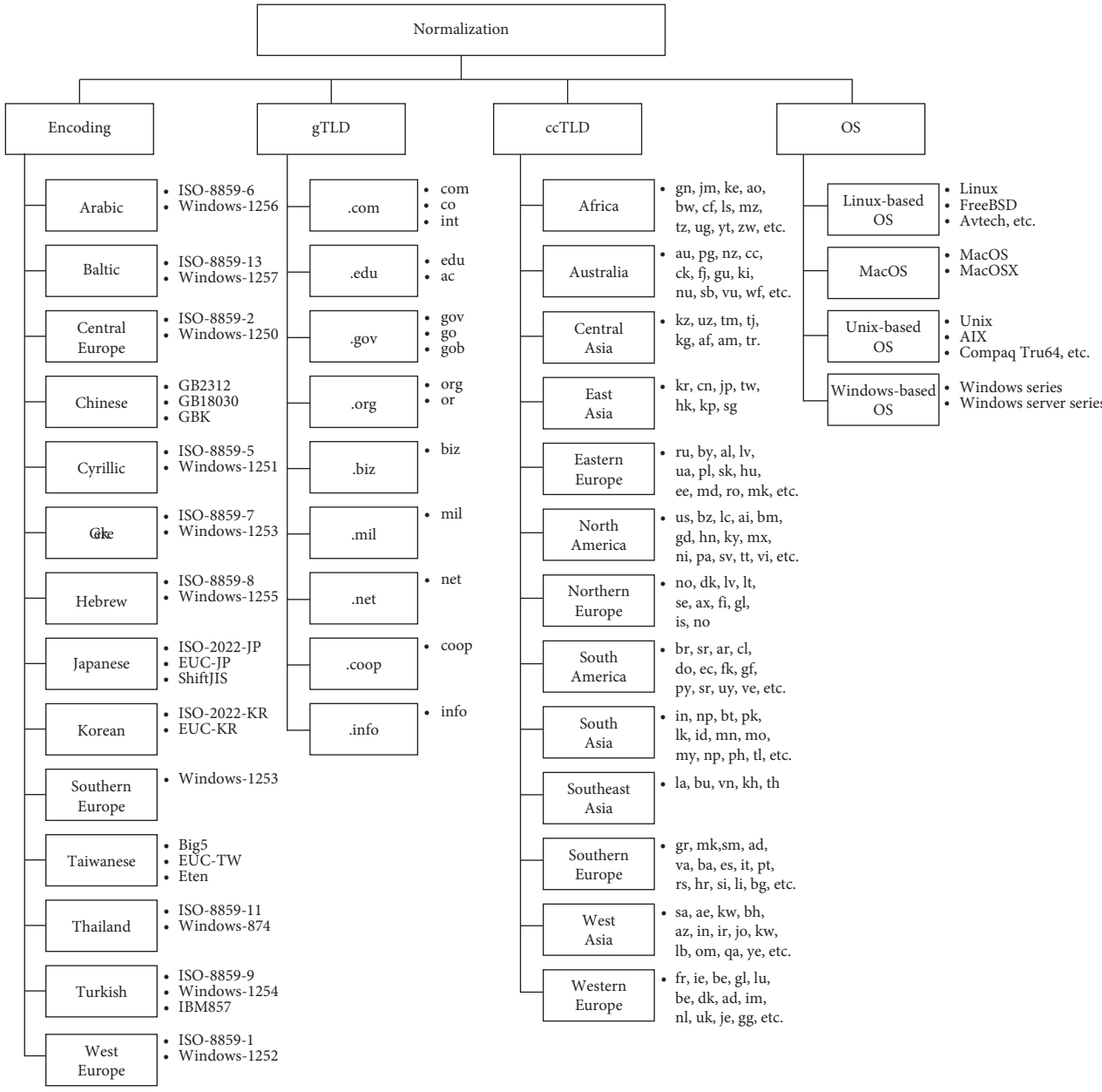


Figure 5: Normalization of each feature elements.

$$\text{Similarity score} = \sum_{i=1}^{\text{cv}} \left[\text{distance}\left(\text{RC}_{\text{cv}}, \text{TC}_{\text{cv}}\right) \times \text{weight}_{\text{cv}}\right],$$

cv: case vector (i.e., encoding, IP address, domain, date, and OS). (1)

There are various approaches to set the weight of the case vector, such as the heuristic method, logistic regression analysis, and attribute weighting methods. Furthermore, these weight values need to be periodically updated to be applied to the study of recent attack trends. However, for the initial setting, it is difficult to set the exact numerical value for each weight values in accordance with the case vector. In our experiment, we set the impact and the weight of the case vector as high, medium, and low according to their importance so that to concretely categorize the attacker and the victim. Above all, since encoding makes it possible to infer the static located information of the attacker, we defined encoding as high-quality information. IP address and domain were defined as medium-quality information. These case vectors enable the identification and specification of the victim. Finally, the targeted date and OS were defined as low-quality information. To measure clustering and similarity, all values of the case vector mixed as numbers and letters were normalized to have a value from 0 to 1. Obviously, since these values can be subjective, in order to prevent this subjective bias, these values should be acquired and thoroughly reviewed by several experts. This technique can be easily applied using expert knowledge of investigation experts and is easy to understand from researchers' viewpoint. The quantitative method for setting and

Input: $TCs(Tested\_DB)$/* The Tested_DB indicates the cases-centric DB */
RC (*Retrieved_Case*) ⟵ $\{Encoding_{RC}, IP_{RC}, Domain_{RC}, Date_{RC}, OS_{RC}\}$/* RC means one of the retrieved cases. */
W (Weight) ⟵ $\{Encoding_W, IP_W, Domain_W, Date_W, OS_W\}$
Output: *Similarity_score*

(1) TC$\{Encoding_{TC}, IP_{TC}, Domain_{TC}, Date_{TC}, OS_{TC}\}$ ⟵ TCs
(2) **While** *RC in TCs* **do**
(3) **if** $Encoding_{RC} == Encoding_{TC}$ **then**
(4) *Encoding_similarity_value* ⟵ 1.0
(5) **else**
(6) *Encoding_similarity_value* ⟵ 0.0
(7) **end**
(8) IP_{RC} $\{Octet\_A_{RC}, Octet\_B_{RC}, Octet\_C_{RC}, Octet\_D_{RC}\}$, IP_{TC} $\{Octet\_A_{TC}, Octet\_B_{TC}, Octet\_C_{TC}, Octet\_D_{TC}\}$
(9) **if** $(Octet\_A_{RC} == Octet\_A_{TC}) \| (Octet\_B_{RC} == Octet\_B_{TC}) \| (Octet\_C_{RC} == Octet\_C_{TC}) \| (Octet\_D_{RC} == Octet\_D_{TC})$ **then**
(10) *IP_similarity_value* ⟵ 1.0
(11) **else if** $(Octet\_A_{RC} == Octet\_A_{TC}) \| (Octet\_B_{RC} == Octet\_B_{TC}) \| (Octet\_C_{RC} == Octet\_C_{TC})$ **then**
(12) *IP_similarity_value* ⟵ 0.75
(13) **else if** $(Octet\_A_{RC} == Octet\_A_{TC}) \| (Octet\_B_{RC} == Octet\_B_{TC})$ **then**
(14) *IP_similarity_value* ⟵ 0.5
(15) **else if** $(Octet\_A_{RC} == Octet\_A_{TC})$ **then**
(16) *IP_similarity_value* ⟵ 0.25
(17) **else**
(18) *IP_similarity_value* ⟵ 0.0
(19) **end**
(20) $Domain_{RC}$ $\{ServiceName_{RC}, gTLD_{RC}, ccTLD_{RC}\}$, $Domain_{TC}$ $\{ServiceName_{TC}, gTLD_{TC}, ccTLD_{TC}\}$
(21) **if** *an identical domain* **then**
(22) *Domain_similarity_value* ⟵ 1.0
(23) **else if** $(ServiceName_{RC} == ServiceName_{TC}) \| (gTLD_{RC} == gTLD_{TC}) \| (ccTLD_{RC} == ccTLD_{TC})$ **then**
(24) *Domain similarity_value* ⟵ 0.8
(25) **else if** $(gTLD_{RC} == gTLD_{TC}) \| (ccTLD_{RC} == ccTLD_{TC})$ **then**
(26) *Domain_similarity_value* ⟵ 0.3
(27) **else if** $(ServiceName_{RC} == ServiceName_{TC})$ **then**
(28) *Domain_similarity_value* ⟵ 0.1
(29) **else if** $(ccTLD_{RC} == ccTLD_{TC})$ **then**
(30) *Domain_similarity_value* ⟵ 0.1
(31) **else if** $(gTLD_{RC} == gTLD_{TC})$ **then**
(32) *Domain_similarity_value* ⟵ 0.1
(33) **else**
(34) *Domain_similarity_value* ⟵ 0.0
(35) **end**
(36) $Date\_variance \longleftarrow |Date_{RC} - Date_{TC}|$/* It converts a date format year, month and day (i.e., yyyy-mm-dd) into a day calculated with numeric. */
(37) **if** $0 \le Date\_variance \le 365$ **then**
(38) *Date_similarity_value* ⟵ 1.0
(39) **else if** $365 < Date\_variance \le 1095$ **then**
(40) *Date_similarity_value* ⟵ 0.75
(41) **else if** $1095 < Date\_variance \le 1825$ **then**
(42) *Date_similarity_value* ⟵ 0.5
(43) **else if** $1825 < Date\_variance \le 2555$ **then**
(44) *Date_similarity_value* ⟵ 0.25
(45) **else if** $2555 < Date\_variance$ **then**
(46) *Date_similarity_value* ⟵ 0.0
(47) **end**
(48) **if** $OS_{RC} == OS_{TC}$ **then**
(49) *OS_similarity_value* ⟵ 1.0
(50) **else**
(51) *OS_similarity_value* ⟵ 0.0
(52) **end**
(53) $Similarity\_score \longleftarrow (Encoding\_similarity\_value \times Encoding_W) + (IP\_similarity\_value \times IP_W) + (Domain\_similarity\_value \times Domain_W) + (Date\_similarity\_value \times Date_W) + (OS\_similarity\_value \times OS_W)$
(54) **return** *Similarity score between RC and TC*
(55) **end while**

Algorithm 1: Similarity measure module.

updating the weight value is an issue worth addressing in further research. In the present study, we set the weight values for the case vector including the encoding, IP address, domain, attack date, and OS (see Table 2).

Some case vectors' distance cannot be directly estimated, as they have mixed numerical and nominal data (such as IP address range and domain name). For this reason, to calculate the distance between the nominal data, we defined the discrete similarity measure. The similarity of IP addresses was calculated by measuring the similarity among the same octet of two given IP addresses. The IP address space is composed of a number combination of four octets separated by ".". In the present study, we compared if octets from the 1st octet to the 4th octet of RC and TC were identical. Subsequently, a similarity value was assigned to the IP address vector. We suggested the discrete similarity value between two IP addresses as visible in Table 2. The proposed approach is advantageous in that it enables the distance calculation between the IP addresses efficiently:

(i) IP address of RC: zzz: yyy: xxx: www

(ii) IP address of TC: zzz: yyy: xxx: www

Meanwhile, the similarity between domains is calculated according to their domain properties. The domain is composed of the gTLD, ccTLD, and service name. The gTLD refers to a generic top-level domain in the domain rule. For instance, .com and .co are used for commercial companies or organizations, .org and .or are used for nonprofit organizations. .go and .gov are used for government and state agencies. Besides, ccTLD refers to a country code top-level domain in the domain rule and means a unique sign that represents a specific region, such as .kr, .cn, .br, and .uk. DNS makes change in the IP address into a unique Domain Name which is easy to remember because it consists of a combination of an alphabet letter and a number. Among the Domain Name, the service name is built corresponding with the characteristics of the groups, organizations, or corporations that the gTLD is intending and pursuing. The service name has diverse and different names depending on the categories of the gTLD, such as educational institutions, commercial enterprises, military organizations, nonprofit organizations, and government and state agencies. Unlike other case vectors, we set the rule for estimating the similarity of the domain, as depicted in Table 2.

Furthermore, we defined the attack date similarity. Similar to the offline criminal investigation case, if the time of a crime occurrence is near, we can analyse the cases as a similar crime with a cross-analysis of the target, area, and the criminals' patterns. The similarity value depends on the period difference between a new case and existing cases. As visible in Table 2, the similarity value is described according to the date gap of two cases that occurred on different dates. In summary, according to the similarity degree of a variation range of a section, the similarity values of the attack IP address, domain, and attack date were set to the similarity value between 0 and 1.

3.3.2. Clustering Processing. Merely sorting the data and visually analysing them render it difficult for an investigator to infer the correlations and similarity among the potential features of incidents. Hence, an advanced tool that would capture the complex underlying structures and data properties is required. Accordingly, in the present study, we conducted the clustering process using the EM algorithm based on the probability of the individual data attributes. This algorithm does not restrict the number of clusters in the parameters but automatically generates a number of valid clusters by cross-validation. Thereafter, the algorithm determines the probability that some data items existed in the cluster by maximizing the correlation and dependence among the objects. We applied practically the EM algorithm to 80,948 data items having the information of encoding, gTLD, ccTLD, and OS from 212,093 data for clustering. The character encoding was normalized by a group of congenial cover code units (ISO-8859, MS Windows character set, GB, and EUC series). We excluded the Unicode because it is too general, which accounts for the majority of the collected encoding data for clustering. In the case of the service name, even if we can find out similar combinations of alphabet letters or numbers, it is not easy to find commonality or relevance between them. Therefore, it is not suitable for being used as the similarity measure of the reasoning engine. Consequently, characteristics and metadata concerning the 12 clusters were obtained (see Table 3). These clustering results are also visualized and stored in the database (see Figure 6).

The donut charts include the different features from outside to inside (in order), with the corresponding share of each feature value separated by a different colour code within this same circle. Each cluster consists of four circles, and the circle represents, from the outside to the inside, the encoding, gTLD, ccTLD, and OS. The percentage in Table 3 represents how many cases one cluster contains among all of website defacement cases collected from the zone-h.org site. The representative hacker represents a notable hacker or hacking group among the members of them in each cluster. As described in Figure 6, clusters of similar patterns were found in the clusters. The most conspicuously similar clusters were 4 and 7, which had the feature of using Arabic and Chinese, a feature of the attack against an industrial organization whose headquarters are located in Western Europe. The cases in Clusters 4 and 7 accounted for 41.29 percent among all of website defacement cases collected from the zone-h.org site. The results of the clustering process contribute to the concretization of the similarity between the new and existing cases. A large number of new cases have flowed in the database, and then, if the clustering process is performed with the dataset, a clustering result may take on a different pattern, of course.

4. Application

4.1. Experimental Results and Analysis. Considering that the assumption that the attackers tend to use similar or unique attack methods is not always valid, and it is difficult to evaluate the accuracy of the similarity mechanism. As time progresses, attackers' hacking skills advance, and in addition, the attack plan, campaign purpose, and target groups can change depending on the situation. Therefore, in the present

Table 2: Value and the weight for the similarity score by the case vector. All of the values of the similarity score are normalized to 0 or 1.

Case vector	Weight	Impact	The similarity measure between a new case and existing cases	Value
Encoding	0.5	High	—	0 or 1
IP address	0.2	Medium	If the same (e.g., 143.248.1.6 and 143.248.1.6)	1
			If the 1st, 2nd, and 3rd octet are matched (e.g., 143.248.1.6 and 143.248.1.8)	0.75
			If the 1st and 2nd octet are matched (e.g., 143.248.1.6 and 143.248.4.4)	0.5
			Only the 1st octet is matched (e.g., 143.248.1.6 and 143.13.2.4)	0.25
			No common octet (e.g., 143.248.1.6 and 163.13.2.5)	0
Domain	0.15	Medium	An identical domain	1
			Service name is matched, and one of the gTLD and ccTLD is matched	0.8
			gTLD and ccTLD is matched	0.3
			Service name is matched	0.1
			ccTLD is matched	0.1
			gTLD is matched	0.1
			Nonidentical domain	0
Date	0.1	Low	Period of about 6 months back and forth (1 year)	1
			Period of about 18 months back and forth (3 years)	0.75
			Period of about 30 months back and forth (5 years)	0.5
			Period of about 42 months back and forth (7 years)	0.25
			Over period of about 42 months (over 7 years)	0
OS	0.05	Low	—	0 or 1

Table 3: Characteristics and metadata of several different clusters derived from the clustering processing.

Cluster number	Ratio (%)	Description	Representative hacker (group)
0	7.84	The group uses Central European languages. They principally attacked against the profit organization and Linux-based OS in Western Europe.	JaMaYcKa, Super2li
1	8.16	The group uses Arabic and Cyrillic. They principally attacked against the organization that manages the network and Linux-based and Unix-based OS. Their attack region is distributed throughout Southern Europe, South America, Eastern Europe, and Southeast Asia.	BI0S
2	10.36	The group uses Central European languages. They principally attacked against the organization that manages the network and nonprofit organizations in Western Europe.	JaMaYcKa
3	9.33	The group uses Central European languages. They principally attacked against the profit organization and Windows-based OS in Western Europe.	1923Turk
4	25.36	The group uses Arabic and Chinese. They principally attacked against the profit organization and Windows-based OS in Western Europe.	EL_MuHaMMeD, federal-atack.org
5	1.73	The group uses Central European languages. They principally attacked against the profit organization and Unix-based OS in Southern Europe and Eastern Europe.	d3b~X, SuSKuN
6	5.24	The group uses Central European languages. They principally attacked against the profit organization, the educational institution, the government and state agencies, and also Windows-based OS in East Asia.	1923Turk

Table 3: Continued.

Cluster number	Ratio (%)	Description	Representative hacker (group)
7	15.93	The group uses Arabic, Chinese, and Turkish. They principally attacked against the profit organization and Linux-based OS in Western Europe.	Rya, iskorpitx
8	9.11	The group uses Central European languages. They principally attacked against the profit organization and Windows-based OS in Western Europe.	1923Turk
9	3.63	The group uses Central European languages. They principally attacked against the profit organization and Linux-based OS in South America and Eastern Europe.	Hmei7
10	1.39	The group uses Central European languages. They principally attacked against Windows-based OS in South America and Southeast Asia. Their attack target is mostly the educational institution and the government and state agencies.	BHS, F4keLive
11	1.92	The group uses Arabic and Central European languages. They principally attacked against the profit organization and Windows-based OS in Southern Europe.	EL_MuHaMMeD, linuXploit_cre

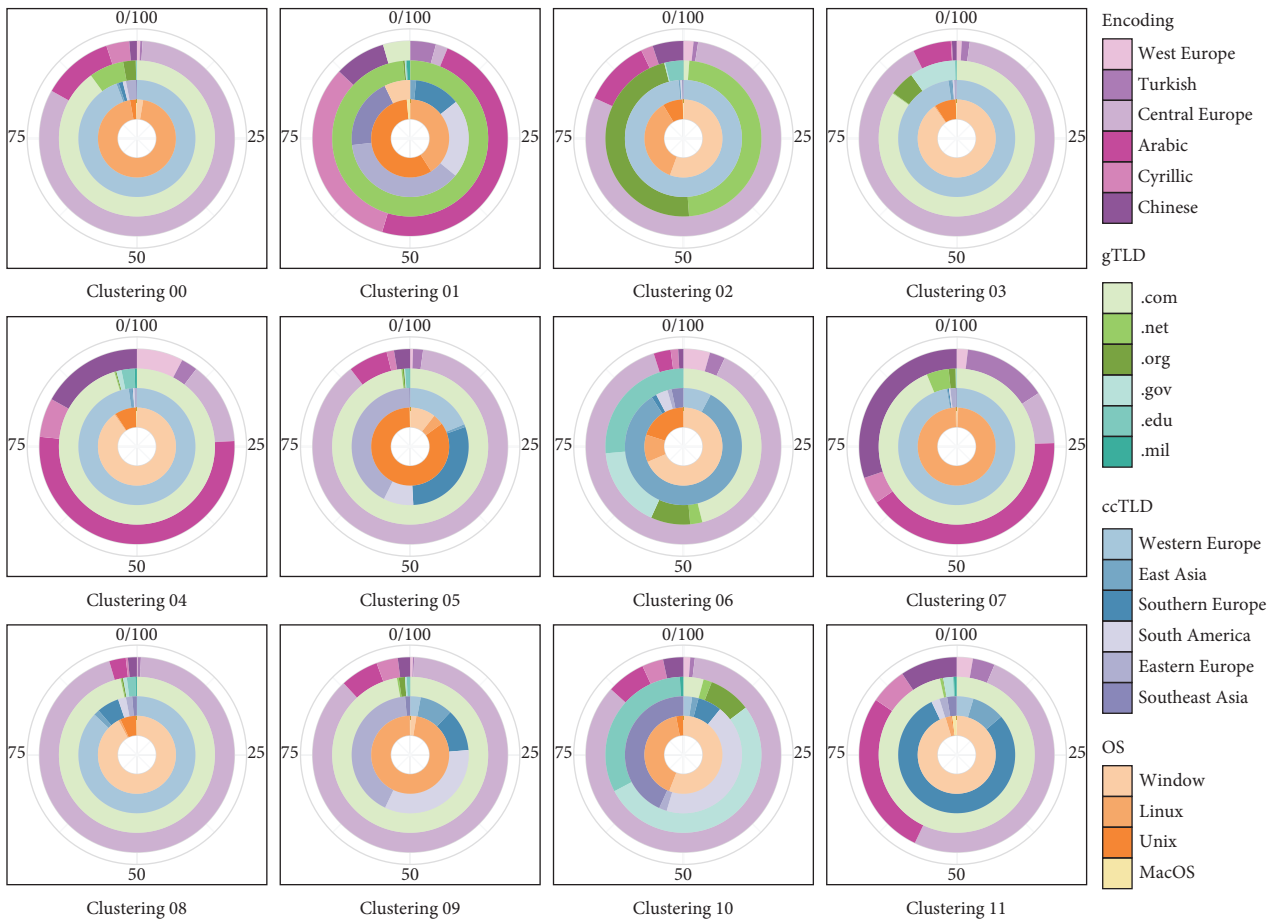


Figure 6: Visualization of the 12 different clusters (00 through 11) in our data annotated with various features: encoding, gTLD, ccTLD, and OS and their corresponding share (legend on the right side).

study, rather than evaluating the accuracy of the similarity mechanism, we tested the overall performance of the proposed methodology with the ratio of correctly identified hackers. The developed testing procedures unfolded in the following four steps and are depicted in detail in Figure 7: where "K" presents all hackers within the database.

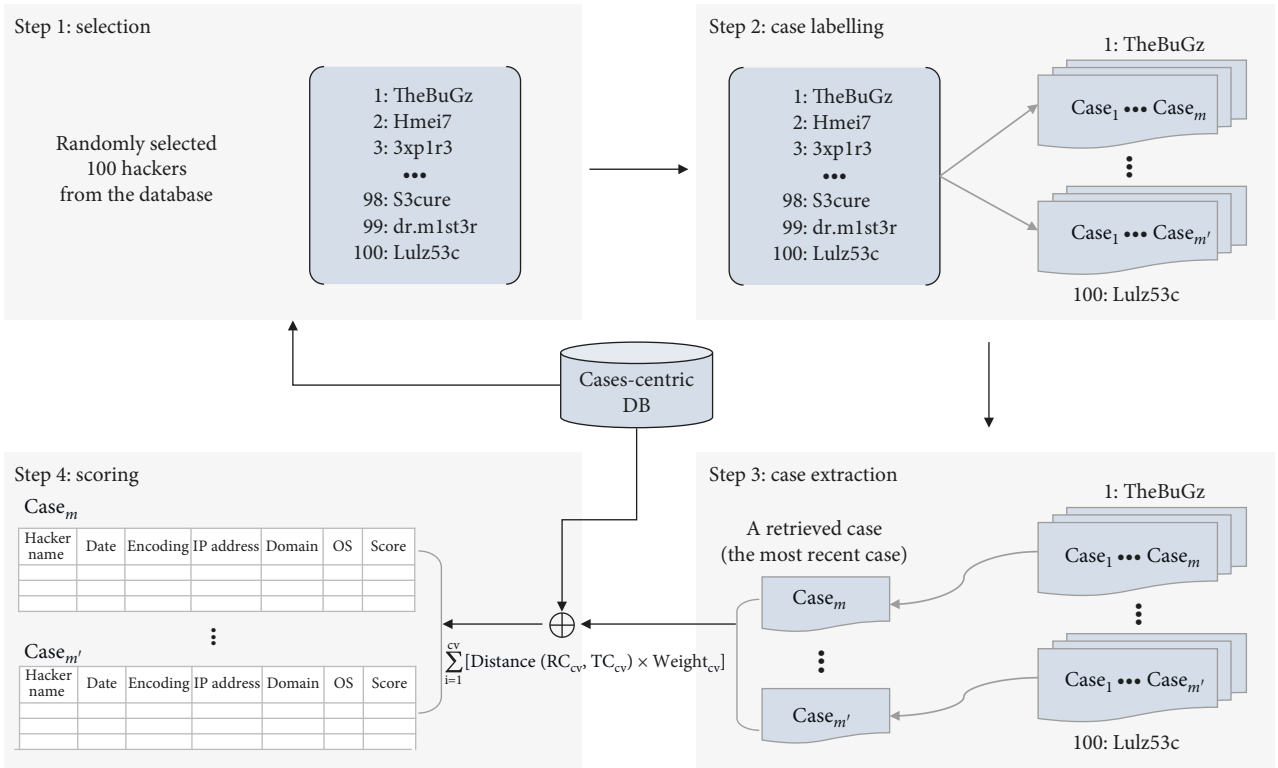


Figure 7: The developed testing procedures from step 1 to step 4.

$$R_k = \frac{\text{Count}\left(\text{Cases}_k^m\right)}{\text{Count}\left(\text{Cases}_k^{\text{all}}\right)}, \tag{3}$$

where "m" means the past cases which are within the defined scope concerning a randomly selected hacker "k."

(i) Step 1, selection: the measurement objects, i.e., 100 hackers were randomly selected from the database.

(ii) Step 2, case labelling: we retrieved all previous attack cases conducted by the randomly selected 100 hackers in Step 1 and then subsequently labelled all previous attack cases by each hacker name.

(iii) Step 3, case extraction: we selected the most recent case among the cases labelled in Step 2 as an input value. The similarity score was then estimated by comparing the most recent case (i.e., RC—one of the retrieved cases) with all other cases in the database (i.e., TCs—all cases in the cases-centric DB).

(iv) Step 4, scoring: similarity score was sorted depending on the value and the weight for the similarity score by the case vector (see Table 2), in the descending order. Whenever the similarity value was 0, it was not displayed on the scoring list of Step 4. The feasibility of the proposed methodology was evaluated based on how many past cases of a hacker there were in the N scope at the scoring list of Step 4; that is, regarding the ratio of the attack cases by each hacker, we checked whether the cases were included at the top N scope (N scope: from the top 1 percent to the top 30 percent):

$$N_{\text{Scope}} = \frac{\text{Count}\left(\text{Cases}_K^{\text{Scope}}\right)}{\text{Count}\left(\text{Cases}_K^{\text{all}}\right)} \times 100, \tag{2}$$

First, we randomly picked 100 hackers from the collected dataset (i.e., cases-centric DB); thereafter, we retrieved and extracted all past attack cases for each hacker. The extracted past cases were labelled with the hacker's name. Figure 8 depicts the number of website defacement attack cases in the past for each hacker. In Steps 3 and 4, similarity between a retrieved case (i.e., the most recent case) and all other stored website defacement cases were measured.

Specifically, we checked whether the result (i.e., the sorted hacker's past cases with a high similarity score) stemming from the similarity measurement was included at the top N scope. This process was meant to check, based on the similarity score, how many past attack cases of randomly picked 100 hackers were included in the defined top N scope. To this end, we divided the top N scope into eight criterion factors from the top 1 percent to the top 30 percent and the ratio R all the past attack cases for each hacker into six criterion factors from 50 percent to 100 percent (i.e., at 10 percent intervals). As illustrated in equations (2) and (3), the N scope and the ratio R were categorized as ratios according to the defined measure rule. More specifically, the criterion of the top N scope, i.e., "top N percent" was based on the result derived from the similarity measurement. Attack cases were sorted in order of high similarity score, and therefore, the cases were within the range of top N scope (see Figure 9). Also, in the case of the hacking case ratio of a randomly

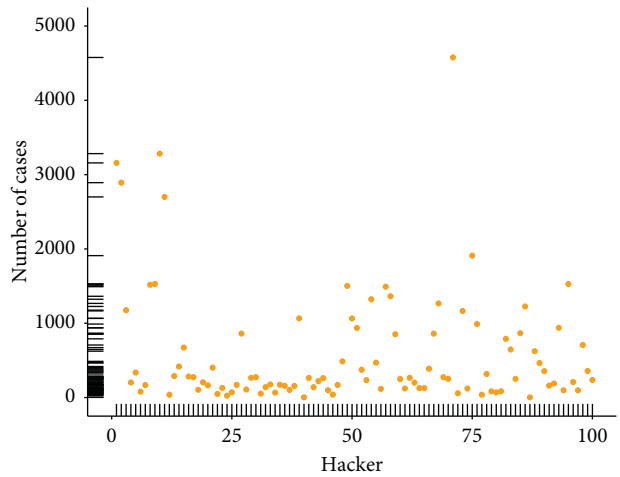


Figure 8: The number of website defacement attack cases in the past of each hacker.

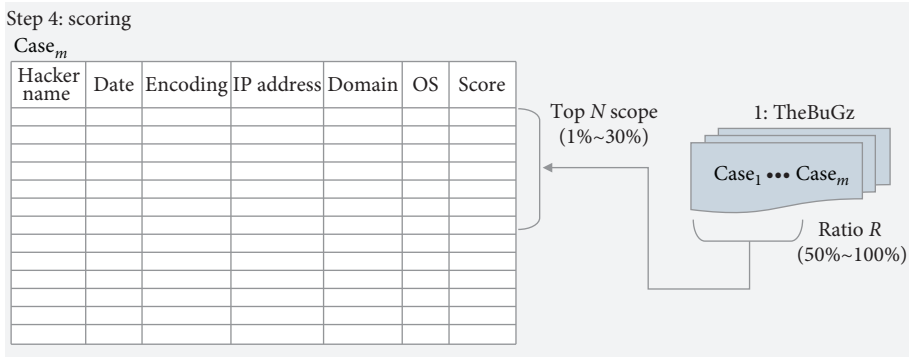


Figure 9: Scoring step on the top N scope and the ratio R.

selected hacker, some parts of the past attack cases (i.e., ratio R) concerning a hacker were within the defined N scope (see Figure 9).

Figure 10 shows the number of an identified hacker for a retrieved case (i.e., the most recent case) among all hacking cases of each hacker. The X-axis in Figure 10 shows the criterion of the top N scope including the eight criterion factors (%) and of the ratio R including the six criterion factors (%). The Y-axis presents the number of an identified hacker in the top N scope among the randomly selected 100 hackers in Step 1. As can be seen in Figure 10, the higher the ratio R and the narrower the N scope, the lower the number of an identified hacker in the top N scope among the randomly selected 100 hackers. On the other hand, the lower the ratio R and the wider the N scope, the higher the number of identified hackers in the top N scope among the randomly selected 100 hackers. Consequently, even if hacking cases were caused by the same hacker, as the hacker or hacking group which only attacked the same or similar objects were rare, it is impossible to draw results with a high similarity score for all cases of a hacker. Nevertheless, the results demonstrated that the proposed CBR-based decision support methodology can successfully reduce the number of hackers and their cases and suggest potential top N percent candidates among hundreds of thousands of cases.

Therefore, an investigator should consider the availability and flexibility of data with respect to the data selection criteria for the similarity measurement. As mentioned above, when a new attack occurs, they can limit the search range of the data and determine the direction of the criminal investigation. With such the reduction in the number of candidate-related cases, the outcomes of our similarity mechanism are highly valuable in terms of reducing the investigation time to determine the potential suspect of a given hacking incident.

4.2. Case Study. As mentioned above, the accuracy of the CBR depends on the quality of the collected data, and the overall accuracy is difficult to evaluate. Nevertheless, although the data are insufficient to evaluate the proposed methodology, the DS and SPE cases include the ground-truth data with specific information related to the hacker or hacking groups. Based on the public ground-truth data of the DS and SPE cases, we found the most similar top three hackers or hacking groups to them and noticed their characteristic, by the proposed similarity measure and the clustering processing.

The hackers of the DS cyberattack defaced the groupware homepage of LG U$^+$, the 3rd largest telecommunication company in South Korea and the English version of the

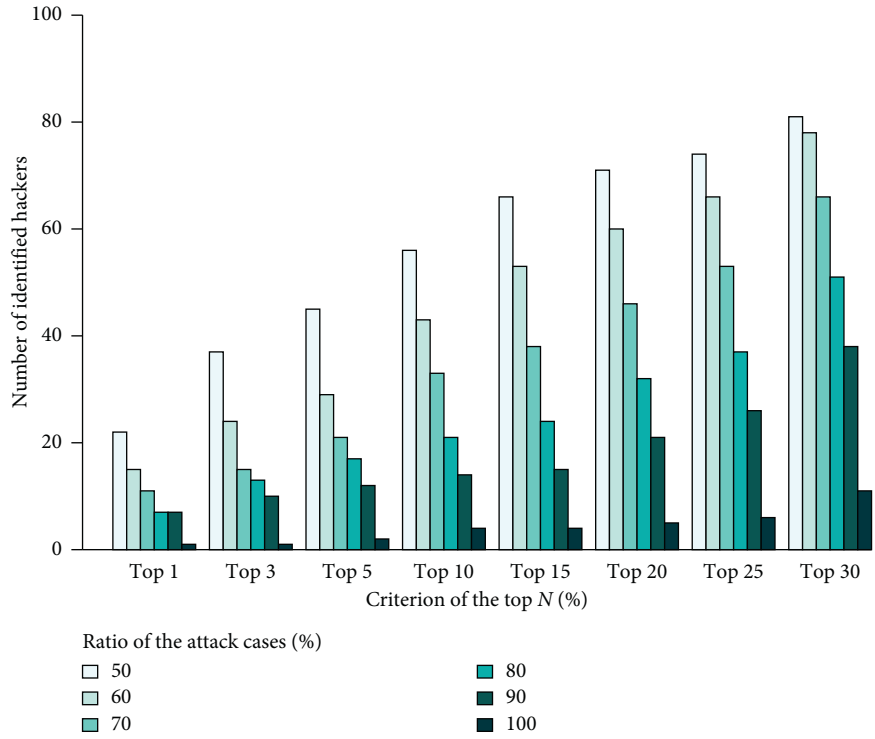


Figure 10: The number of identified hackers in the top N scope among the randomly selected 100 hackers.

Korean Broadcasting System (KBS) homepage. They left unique images and many messages on the defaced websites. The three Calaveras image (i.e., skull image) used in the LG U$^+$'s defaced website appeared on many European websites. The character encoding set of the message was the Western European language system. Based on these insights, we could infer that the hackers' background is European. "HASTATI" was the word written on the KBS homepage, meaning the forefront line of the Roman troops, hinting that the DS cyberattack could be a starting point; rather than a transient attack, it was a persistent one. Even if we excluded other images and messages, as well as other features from the similarity processes due to the unanticipated loss or absence of data, one could establish the similarity and intent of the attackers with reasonable confidence. However, given the sufficiently large hacker profiling source, such abundant data could support and enhance the accuracy of inference. Figure 11 shows the screenshots of the defaced websites at that time.

In the SPE case, similarly to the DS case, some images and messages were left on the computers of SPE. Regarding colour, skulls image, and misspellings, the images Figure 11(c) used in the SPE cases took on the characteristics similar to those of the images Figure 11(b) used in the DS cases. As shown in Figure 11, the colour schemes in green and red and the visual similarities seen in skull image are other crucial elements for crime tracing. In both the DS and SPE cases, the phrase such as "this is the beginning" and "your data" were commonly found in the messages. However, given the intentional hacking nature of forging or hiding their identity, motivation, and location, some experts say that these characteristics are not the conclusive proof that Sony has been attacked by the same hacker [49–51].

For the evaluation of the results of the case study, we first measured the similarity between the new website defacement cases (i.e., the DS and SPE cases) and the collected existing cases in the database. This approach coheres with the CBR process used in cybercrime investigation (see Figure 2). Two new website defacement cases, the DS and the SPE were applied as RC, and the similarity score for each of these two cases was computed using the similarity measure (see equation (1)) proposed in Section 3.3.1. Provided that, because the DS and SPE cases do the function of the target cases as an input value, we considered a direct comparison between the DS and SPE cases for the similarity score was not appropriate [52].

The similarity measure mentioned in the previous paragraph is based on the metadata released by an analysis report of the DS and SPE real cases. We summarized further the characteristics and metadata associated with them in Table 4. The similarity score was derived through comparison between the presented metadata of the DS and SPE cases and all cases in the cases-centric DB. We gave the most similar top three cases among the result of the similarity score (see the right side in Table). Notifier Hmei7 and d3b_X are among the cases that belonged to Clusters 0 and 8, which were the two clusters that exhibited identical characteristics. It can thus be understood that they used the encoding system pertinent to Central European languages based on the Latin language system and typically launched attacks against a profit organization located in Western Europe. Notifier oaddah, M@TRiX, and EL_MuHaMMeD were all classified

(a) (b) (c)

Figure 11: A snippet of website defacement cases by a comparison of examples of the DS and SPE: the defaced LG U$^+$ groupware homepage (a) and KBS homepage (b) in the DS case and the defaced website in SPE case (c).

Table 4: Further characteristics and metadata associated with the DS and SPE cases.

	Retrieved case	Tested cases		
	Case name		Notifier	
	DarkSeoul (DS)	Hmei7	d3b_X	StifLer
Encoding	**Windows-1252**	Windows-1252	Windows-1252	ISO-8859-9
IP address	**203.248.195.178**	203.86.238.68	203.124.37.66	77.92.108.3
Domain	gyunggi.onnet21.com	http://www.garycheng.com	health.ajk.gov.pk	yapikimyasallari.com.tr
Date	**20 Mar 2013**	6 Feb 2014	4 Feb 2014	8 Jun 2013
OS	**Windows**	Windows	Windows	Windows
Similarity	—	0.690	0.675	0.665
Cluster	—	0	8	4
	Retrieved case	Tested cases		
	Case name		Notifier	
	Sony pictures Entertainment (SPE)	Oaddah	M@TRiX	EL_MuHaMMeD
Encoding	**EUC-KR, EUC-CN**	GB2312	GB2312	GB2312
IP address	**203.131.222.102**	203.124.15.55	208.29.19.8	208.116.45.34
Domain	http://www.sonypicturesstockfootage.com	http://www.hzkcgg.com	dax.digitalrom.com	digitalairstrip.net
Date	**24 Nov 2014**	14 Jun 2012	16 Dec 2002	18 June 2009
OS	**Windows**	Windows	Windows	Windows
Similarity	—	0.615	0.615	0.600
Cluster	—	7	7	7

The metadata are arranged according to the defined case vector, corresponding with the DS and SPE cases on the left side (shown in part in boldface type).

as the same cluster (Cluster 7), where the hackers of Cluster 7 used the encoding system pertinent to Arabic and Chinese languages and typically attacked against the profit organization located in Western Europe.

Next, to ensure the objectivity of the similarity score based on the case study by the DS and SPE, we computed the similarity score of any randomly selected pair from the whole case. Figure 12(a) shows the distribution of the similarity score of the randomly selected cases. We took the distribution of the similarity score using the central limit theorem, which describes the average distribution of random samples extracted from a finite population. The distribution shows that the calculation of the similarity score of the randomly selected two website defacement cases was repeatedly performed for 10,000 times. The similarity scores of any randomly selected pair of cases were typically distributed around 0.3. This result (Figure 12(a)) substantiates that the similarity scores are not low, even if the similarity scores of the DS and SPE cases (Figure 12(b)) do not appear numerically high. Figure 12(b) shows the similarity scores of the DS and SPE cases. The top score of the similarity was 0.69 in the DS case, and all measured cases concentrated around the similarity score (X-axis) of 0.0 to 0.15 and of 0.5 to 0.6. In the SPE case, the top score of the similarity was 0.615, and all measured cases concentrated around the similarity score (X-axis) of 0.0 to 0.2.

Figure 13 shows the distribution of the similarity score for randomly selected 100 hackers mentioned in Section 4.1. To know the mean value of the similarity score for each hacker case, we calculated the similarity score from the hacker's own past cases. Cases used for the similarity score means not all cases in the cases-centric DB but just the past cases conducted by the hacker in the cases-centric DB. The mean value of the similarity scores in the hackers is 0.5233. The similarity scores of the tested cases in Table 4 is above the mean value. Thus, the similarity scores for each hacker adequately underpin the similarity scores from the TCs in DS and SPE.

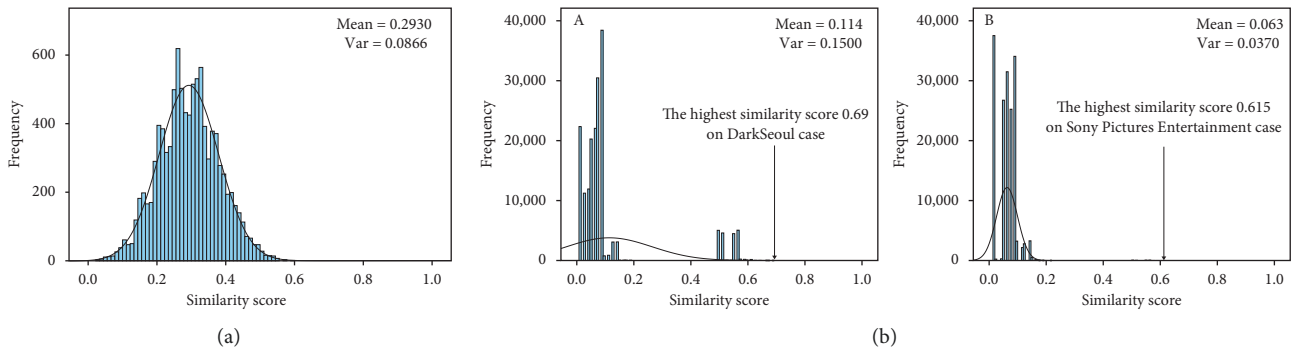


Figure 12: (a) Probability distribution of the similarity score for any pair of randomly selected cases; (b) distribution of the similarity value between the collected website defacement cases with the DS case (A) and the distribution of the similarity value between the collected website defacement cases with the SPE case (B). The similarity was calculated between each studied case and all other cases in our system.

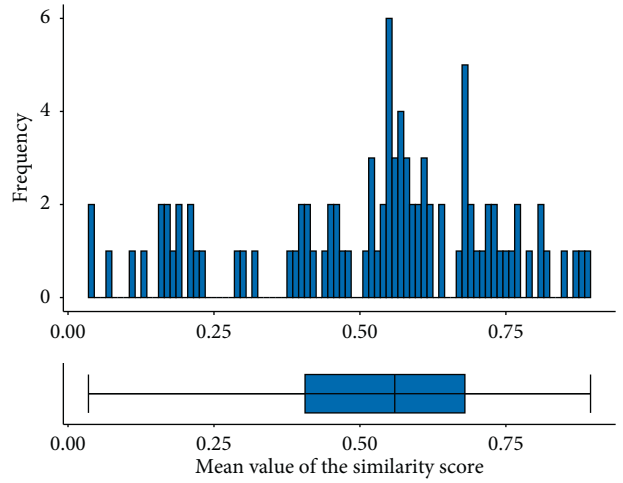


Figure 13: Distribution of the similarity score for randomly selected 100 hackers.

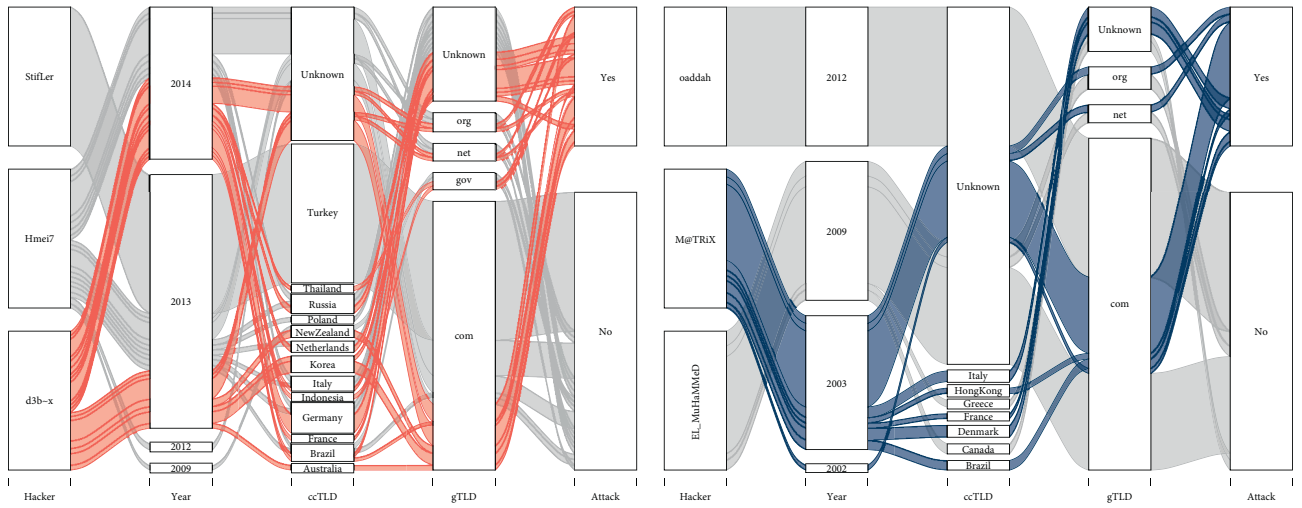
4.3. Follow-Up Investigation. A case study is a research method involving an in-depth and detailed investigation of a subject of study, as well as its related contextual methodology. Hence, we conducted follow-up investigations of the most similar top three hackers, as mentioned above in Table 4. According to the results, specifically, over 93 percent of the hacker's attacks were similar to the DS case that occurred in 2013 and 2014. Their major targets were .com domain sites, and they targeted primarily Germany, Italy, New Zealand, Russia, Turkey, Taiwan, and South Korea (see Table 5). Two hackers (i.e., Hmei7 and d3b_X) primarily attacked government agencies. Interestingly, 20 percent of the attacks by the hackers named d3b_X targeted South Korea. In the SPE incident, the similar hacker's attacks occurred throughout the period from 2002 to 2014. The hackers named M@TRiX and EL_MuHaMMeD intensively executed such attacks in 2003 and 2009. Their major targets were .com (or .co) and .org domain sites, and they targeted primarily Brazil, Canada, Denmark, France, Greece, Hong Kong, and Italy (see Table 5). Two hackers (i.e., M@TRiX and EL_MuHaMMeD) primarily attacked commercial agencies and additionally attacked the public and network agencies. As shown in Figure 14, to describe the follow-up investigation more discernibly and to focus on the attack flow, we used an alluvial diagram, which is a type of Sankey diagram developed to represent changes in a network structure over time [53]. It shows the investigation of the top three hackers with website defacement cases most similar to the DS case and SPE case. The case vectors were based on the attack year, ccTLD, and gTLD. The thickness of the attack flow in this figure means the degree of attack. This network visualization method could support an investigator to understand the flow and core of the crime clearly, by listing the multidimensional evidence that is complicatedly entangled or hidden, such that it does not look presentable.

5. Limitations and Discussion

The CBR algorithm has the disadvantage that the performance evaluation may be degraded if the property describing the case is inappropriate. Therefore, in order to obtain more accurate results, cross-data analysis with other various data sources should be considered. For example, cybercrime statistics data from law enforcement agencies, threat intelligence data from malware analysis groups, and vulnerability databases could be useful resources to

Table 5: Follow-up investigation on the top three hackers with website defacement cases most similar to the DS case and SPE case. The case vector value means the hacker's attack rate.

Domain	DS case			SPE case		
	Hmei7	d3b_X	StifLer	Oaddah	M@TRiX	EL_MuHaMMeD
Com	78.32	85.81	100.00	100.00	86.27	82.98
Edu	1.62	0.96	—	—	1.76	1.91
Net	3.40	3.20	—	—	5.46	5.74
Gov	12.16	6.51	—	—	1.06	—
Year	Hmei7	d3b_X	StifLer	Oaddah	M@TRiX	EL_MuHaMMeD
2002	—	—	—	—	10.74	—
2003	—	—	—	—	89.08	—
2006	—	—	—	—	—	—
2007	0.09	—	—	—	0.18	—
2008	—	—	—	—	—	—
2009	3.15	—	—	—	—	99.57
2010	0.09	—	—	—	—	—
2011	0.34	—	—	—	—	—
2012	3.40	—	—	100.00	—	—
2013	34.86	39.17	100.00	—	—	—
2014	58.08	59.77	—	—	—	0.43
2015	—	1.07	—	—	—	—



(a) (b)

Figure 14: Follow-up investigation on the top three hackers with website defacement cases that are most similar to the DS case (a) and SPE case (b).

improve the accuracy and usability of our proposed methodology. However, at the time of writing the present paper, we did not have access to open and public data concerning cybercrime.

For that reason, we tried to demonstrate the practicability of the proposed methodology as a proof of concept. Therefore, we focused on the dataset of the zone-h.org that includes a large number of website defacement cases. Although the zone-h.org provides an extensive dataset on the past incident events, not all incidents can be included in our study. Therefore, if a hacker penetrated some target organizations by APT attacks and performed stealthy activities, such hacking activities would not be reported in the dataset of the zone-h.org, and the proposed methodology would not be able to detect similar cases with reasonable confidence.

6. Conclusion and Future Work

In this study, the similarity of website defacement cases was assessed through the similarity measure and the clustering processing using the CBR as a methodology. The collected raw data of the defaced web sites' resources was sanitized via data parsing and data cleaning process. Also, based on the large size of real dataset, data-driven analysis for the hacker profiling is achieved. To this end, the case vector was designed, and the significant features were chosen for applying to the case-based reasoning. For a successful cybercrime investigation, hacker profiling via clustering analysis is the most basic and important process; in order to find out the relevant incident cases and significant data on some prime incidents, data-driven

and evidence-driven decision making should be the critical process. Also, reducing the amount of data and time to be analysed are important factors to deliver the high value of intelligence data.

Although the obtained results appear to be sound and meaningful, it is difficult to evaluate the accuracy of the results unless the attacker is captured. Naturally, the ground-truth data with specific information about the involved hacking groups for verification are rare (i.e., no adversary claimed that the two attacks were the result of their actions). However, it is noteworthy that our methodology provides a meaningful insight into the confidential and undercover network of cybercrime as well, especially when there is a lack of information. Also, the proposed methodology contributes to facilitate the analysis and reducing the time required for searching for possible suspects of cybercrime. We believe that the proposed system is meaningful for further exploration and correlation of various website defacement cases.

As mentioned in Discussion and Limitations, a cross-data analysis with other various data sources should be reviewed. Said differently, the use of additional online or offline information acquired by human intelligence (HUMINT) or different types of signal intelligence (SIGINT) and sources may also help to reason composition requirements of crime and reduce the category of investigation. Furthermore, the proposed methodology can be expanded into incident information for compatibility and information exchangeability with other cyberthreat intelligence system as the Structured Threat Information eXpression (STIX) and Trusted Automated eXchange of Indicator Information (TAXII), which are key strategic elements of the information-sharing system [54].

There are features such as the particular messages (i.e., thanks-to, notifier, nationality, religion, and anniversary) or image and mp3 file in the web resources which are gathered from the zone-h.org site. Although these features are limited to only a small number of hackers of the web resources, in future research, we will try to study a close-knit network among them, such as the hub hacking group, key player, and followers. Furthermore, we also plan to more definitely classify and systemize the hackers' intents using text mining and mood detection techniques. The findings of this prospective study will contribute meaningful insights to trace hackers' behavioural patterns and to estimate their primary purpose and intent.

Data Availability

The web-hacking dataset applied to our paper can be downloaded from the linked site below. http://ocslab.hksecurity.net/Datasets/web-hacking-profiling.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported under the framework of international cooperation program managed by the National Research Foundation of Korea (No. 2017K1A3A1A17092614).

References

[1] S. S. Response, "Swift attackers' malware linked to more financial attacks," 2016, https://www.symantec.com/connect/blogs/swift-attackers-malware-linked-more-financial-attacks.

[2] S. S. Response, "Wannacry: ransomware attacks show strong links to lazarus group," 2017, https://www.symantec.com/connect/blogs/wannacry-ransomware-attacks-show-strong-links-lazarus-group.

[3] K. lab, "Lazarus under the hood," 2018, https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/07180244/Lazarus_Under_The_Hood_PDF_final.pdf.

[4] Operation Blockbuster, "Destructive malware report," 2016, https://www.operationblockbuster.com/wp-content/uploads/2016/02/Operation-Blockbuster-Destructive-Malware-Report.pdf.

[5] D. Martin and SANS Institute InfoSec Reading Room, "Tracing the lineage of DarkSeoul," 2016, https://www.sans.org/reading-room/whitepapers/critical/tracing-lineage-darkseoul-36787.

[6] D. S. C. T. U. T. Intelligence, "Wiper malware threat analysis," 2013, https://www.secureworks.com/research/wiper-malware-analysis-attacking-korean-financial-sector.

[7] R. Sherstobitoff, M. L. Itai Liba, and O. O. T. C. James Walter, "Dissecting operation troy: cyberespionage in South Korea," 2013, https://www.mcafee.com/enterprise/en-us/assets/white-papers/wp-dissecting-operation-troy.pdf.

[8] N. Horton and A. DeSimone, "Sony's nightmare before christmas: the 2014 North Korean cyber attack on Sony and lessons for US government actions in cyberspace," 2018, https://www.jhuapl.edu/Content/documents/SonyNightmareBeforeChristmas.pdf.

[9] I. K. Lee and S. R. Ramsey, *The Korean Language*, State University of New York, Albany, NY, USA, 2000.

[10] V. Benjamin and H. Chen, "Securing cyberspace: identifying key actors in hacker communities," in *Proceedings of the 2012 IEEE International Conference on Intelligence and Security Informatics*, pp. 24–29, Arlington, VA, USA, June 2012.

[11] Y. Lu, X. Luo, M. Polgar et al., "Social network analysis of a criminal hacker community," *Journal of Computer Information Systems*, vol. 51, no. 2, pp. 31–41, 2010.

[12] J.-W. Jang, H. Kang, J. Woo, A. Mohaisen, and H. K. Kim, "Andro-autopsy: anti-malware system based on similarity matching of malware and malware creator-centric information," *Digital Investigation*, vol. 14, pp. 17–35, 2015.

[13] J. W. Jang and H. K. Kim, "Function-oriented mobile malware analysis as first aid," *Mobile Information Systems*, vol. 2016, Article ID 6707524, 11 pages, 2016.

[14] Y. Ki, E. Kim, and H. K. Kim, "A novel approach to detect malware based on api call sequence analysis," *International Journal of Distributed Sensor Networks*, vol. 11, no. 6, Article ID 659101, 2015.

[15] M. L. Han, H. C. Han, A. R. Kang et al., "Web-hacking dataset for the cyber criminal profiling," 2016, http://ocslab.hksecurity.net/Datasets/web-hacking-profiling.

[16] M. L. Han, H. C. Han, A. R. Kang, B. I. Kwak, A. Mohaisen, and H. K. Kim, "WAHP: web-hacking profiling using case-based reasoning," in *Proceedings of the 2016 IEEE Conference*

on Communications and Network Security (CNS), pp. 344-345, Philadelphia, PA, USA, October 2016.

[17] A. Aamodt and E. Plaza, "Case-based reasoning: foundational issues, methodological variations, and system approaches," *AI Communications*, vol. 7, no. 1, pp. 39–59, 1994.

[18] D. M. L. Martins and F. B. D. Lima Neto, "Hybrid intelligent decision support using a semiotic case-based reasoning and self-organizing maps," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, no. 99, pp. 1–8, 2017.

[19] H. K. Kim, K. H. Im, and S. C. Park, "DSS for computer security incident response applying CBR and collaborative response," *Expert Systems with Applications*, vol. 37, no. 1, pp. 852–870, 2010.

[20] J.-B. Lamy, B. Sekar, G. Guezennec, J. Bouaud, and B. Séroussi, "Explainable artificial intelligence for breast cancer: a visual case-based reasoning approach," *Artificial Intelligence in Medicine*, vol. 94, pp. 42–53, 2019.

[21] M. Relich and P. Pawlewski, "A case-based reasoning approach to cost estimation of new product development," *Neurocomputing*, vol. 272, pp. 40–45, 2018.

[22] E. R. Reyes, S. Negny, G. C. Robles et al., "Improvement of online adaptation knowledge acquisition and reuse in case-based reasoning: application to process engineering design," *Engineering Applications of Artificial Intelligence*, vol. 41, pp. 1–16, 2015.

[23] H. K. Kim, S.-K. Kim, and S.-H. Kim, "Decision support system for zero-day attack response," *Applied Mathematics and Information Sciences*, vol. 6, no. 1, pp. 221S–241S, 2012.

[24] G. Horsman, C. Laing, and P. Vickers, "A case-based reasoning method for locating evidence during digital forensic device triage," *Decision Support Systems*, vol. 61, pp. 69–78, 2014.

[25] G. Horsman, C. Laing, and P. Vickers, "A case based reasoning system for automated forensic examinations," in *Proceedings of the PGNET 2011 the 12th Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting*, pp. 26–31, Liverpool, UK, June 2011.

[26] Z. Yin, Y. Gao, and B. Chen, "On development of supplementary criminal analysis system based on cbr and ontology," in *Proceedings of the 2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, vol. 14, Taiyuan, China, October 2010.

[27] A. J. Pinizzotto and N. J. Finkel, "Criminal personality profiling: an outcome and process study," *Law and Human Behavior*, vol. 14, no. 3, pp. 215–233, 1990.

[28] P. Chen and J. Kurland, "Time, place, and modus operandi: a simple apriori algorithm experiment for crime pattern detection," in *Proceedings of the 2018 9th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pp. 1–3, Zakynthos, Greece, July 2018.

[29] C. J. R. Collie and K. Shalev Greene, "Examining modus operandi in stranger child abduction: a comparison of attempted and completed cases," *Journal of Investigative Psychology and Offender Profiling*, vol. 16, no. 2, pp. 91–109, 2019.

[30] V. Benjamin, B. Zhang, J. F. Nunamaker Jr., and H. Chen, "Examining hacker participation length in cybercriminal internet-relay-chat communities," *Journal of Management Information Systems*, vol. 33, no. 2, pp. 482–510, 2016.

[31] V. Benjamin and H. Chen, "Time-to-event modeling for predicting hacker IRC community participant trajectory," in *Proceedings of the 2014 IEEE Joint Intelligence and Security Informatics Conference*, pp. 25–32, The Hague, The Netherlands, September 2014.

[32] K. Veena and K. Meena, "Identification of cyber criminal by analysing the users profile," *International Journal of Network Security*, vol. 20, no. 4, pp. 738–745, 2018.

[33] F. Iqbal, B. C. M. Fung, M. Debbabi, R. Batool, and A. Marrington, "Wordnet-based criminal networks mining for cybercrime investigation," *IEEE Access*, vol. 7, pp. 22740–22755, 2019.

[34] N. Qazi and B. L. W. Wong, "An interactive human centered data science approach towards crime pattern analysis," *Information Processing & Management*, vol. 56, no. 6, p. 102066, 2019.

[35] N. Jain, P. Sharma, R. Anchan et al., "Computerized forensic approach using data mining techniques," in *Proceedings of the ACM Symposium on Women in Research 2016*, pp. 55–60, ACM, New York, NY, USA, 2016.

[36] P. M. Cozens, G. Saville, and D. Hillier, "Crime prevention through environmental design (cpted): a review and modern bibliography," *Property Management*, vol. 23, no. 5, pp. 328–356, 2005.

[37] H. Hassani, X. Huang, E. S. Silva, and M. Ghodsi, "A review of data mining applications in crime," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 9, no. 3, pp. 139–154, 2016.

[38] A. Sharma and S. Sharma, "An intelligent analysis of web crime data using data mining," *International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 2, no. 3, 2012.

[39] S.-T. Li, S.-C. Kuo, and F.-C. Tsai, "An intelligent decision-support model using FSOM and rule extraction for crime prevention," *Expert Systems with Applications*, vol. 37, no. 10, pp. 7108–7119, 2010.

[40] Y.-H. Tseng, Z.-P. Ho, K.-S. Yang, and C.-C. Chen, "Mining term networks from text collections for crime investigation," *Expert Systems with Applications*, vol. 39, no. 11, pp. 10082–10090, 2012.

[41] A. Malathi and S. S. Baboo, "An enhanced algorithm to predict a future crime using data mining," *International Journal of Computer Applications*, vol. 21, no. 1, 2011.

[42] S. Kapetanakis, A. Filippoupolitis, G. Loukas et al., "Profiling cyber attackers using case-based reasoning," in *Proceedings of the 19th UK Workshop on Case-Based Reasoning (UKCBR 2014)*, Cambridge, UK, December 2014.

[43] R. Al-Zaidy, B. C. Fung, A. M. Youssef et al., "Mining criminal networks from unstructured text documents," *Digital Investigation*, vol. 8, no. 3-4, pp. 147–160, 2012.

[44] M. Zulfadhilah, Y. Prayudi, and I. Riadi, "Cyber profiling using log analysis and k-means clustering," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 7, pp. 430–435, 2016.

[45] S. V. Nath, "Crime pattern detection using data mining," in *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops*, pp. 41–44, Hong Kong, China, December 2006.

[46] ITP.net, "Syria, Egypt crises spur escalation of me cyber attacks," 2013, http://www.itp.net/594742-syria-egypt-crises-spur-escalation-of-me-cyber-attack.

[47] A. McEnery and R. Xiao, "Character encoding in corpus construction," in *Developing Linguistic Corpora: A Guide to Good Practice*, Oxbow Books Ltd., Oxford, UK, 2005.

[48] B. Bos, T. Çelik, I. Hickson et al., "Cascading style sheets level 2 revision 1 (CSS 2.1) specification," W3C Working Draft, 2005, http://www.w3.org/TR/CSS21/.

[49] W. Stuckey, "Massive sony breach sheds light on murky hacker universe," 2018, http://america.aljazeera.com/articles/2014/12/24/sony-hacker-universe.html.
[50] S. Gallagher, "Sony pictures malware tied to Seoul, "Shamoon" cyber-attacks," 2018, https://arstechnica.com/information-technology/2014/12/sony-pictures-malware-tied-to-seoul-shamoon-cyber-attacks/.
[51] J. Pagliery, "Sony hack: signs point to North Korea," 2018, https://money.cnn.com/2014/12/05/technology/security/sony-hack-north-korea-employee/index.html.
[52] K. Ketler, "Case-based reasoning: an introduction," *Expert Systems with Applications*, vol. 6, no. 1, pp. 3–8, 1993.
[53] M. Rosvall and C. T. Bergstrom, "Mapping change in large networks," *PLoS One*, vol. 5, no. 1, Article ID e8694, 2010.
[54] OASIS, "STIX/TAXII standards," 2017-2018, https://oasis-open.github.io/cti-documentation/.

Hindawi
Security and Communication Networks
Volume 2019, Article ID 3093809, 11 pages
https://doi.org/10.1155/2019/3093809

Research Article

Session-Based Webshell Detection Using Machine Learning in Web Logs

Yixin Wu,[1] Yuqiang Sun,[1] Cheng Huang,[1] Peng Jia,[2] and Luping Liu[2]

[1]*College of Cybersecurity, Sichuan University, Chengdu, China*
[2]*College of Electronics and Information Engineering, Sichuan University, Chengdu, China*

Correspondence should be addressed to Cheng Huang; opcodesec@gmail.com

Received 2 April 2019; Revised 14 July 2019; Accepted 24 July 2019; Published 22 November 2019

Guest Editor: Sebastian Schrittwieser

Copyright © 2019 Yixin Wu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Attackers upload webshell into a web server to achieve the purpose of stealing data, launching a DDoS attack, modifying files with malicious intentions, etc. Once these objects are accomplished, it will bring huge losses to website managers. With the gradual development of encryption and confusion technology, the most common detection approach using taint analysis and feature matching might become less useful. Instead of applying source file codes, POST contents, or all received traffic, this paper demonstrated an intelligent and efficient framework that employs precise sessions derived from the web logs to detect webshell communication. Features were extracted from the raw sequence data in web logs while a statistical method based on time interval was proposed to identify sessions specifically. Besides, the paper leveraged long short-term memory and hidden Markov model to constitute the framework, respectively. Finally, the framework was evaluated with real data. The experiment shows that the LSTM-based model can achieve a higher accuracy rate of 95.97% with a recall rate of 96.15%, which has a much better performance than the HMM-based model. Moreover, the experiment demonstrated the high efficiency of the proposed approach in terms of the quick detection without source code, especially when it only considers detecting for a period of time, as it takes 98.5% less time than the cited related approach to get the result. As long as the webshell behavior is detected, we can pinpoint the anomaly session and utilize the statistical method to find the webshell file accurately.

1. Introduction

Webshells have become the main threat challenges for protecting the security of websites. According to the weekly safety report issued by National Computer Network Emergency Response Technical Team/Coordination Center of China (CNCERT/CC) in 2019, the number of websites with backdoors is growing almost every week [1]. As a web service-based backdoor program, webshell is installed through vulnerabilities in web applications or weak server security configuration such as SQL injection, the file including and uploading. Webshells with encryption and obfuscation are often used in attacks mostly because they are difficult to detect by WAF and other antivirus software. A hacker can initiate attacks using webshell tools such as Chinese Chopper [2] and achieve quite a few large malicious attacks like data theft, DDoS attacks, and watering hole attacks [3].

When a malicious webshell attack occurs, there are some exceptions as alertness for admin such as files with an abnormal timestamp, high traffic for a user in very short period time, and files including malicious codes. Attackers can also hide webshell logins in fake error pages.

Due to the high usage of webshell in cyberattacks, there has been much previous research in this field. But most researchers focus on the contents of suspicious files [4–6] or POST contents in HTTP requests [7] and thus ignore features in a sequence of web server logs. With the development of encryption and obfuscation technology [8, 9], it is quite difficult to detect webshell in thousands of website source files, but when we only need to deal with the sequence of several fields in the web logs without considering complex text processing, the problem becomes much simpler. This paper focuses on providing a new webshell detection method based on user's sessions in web logs to improve the accuracy

of webshell detection simply and effectively. Moreover, if the website defenders can identify malicious webshell immediately, they can prevent related cyber threats.

This paper presents a comprehensive framework including log data collection, feature extraction, session identification, and comparison of two machine learning methods. The main innovations of this paper can be summarized as follows:

(i) The presented model utilized a session-based method to detect webshell which is simpler and more efficient than existing methods. Session identification can be more precise using a statistical method which roughly calculates the time interval of each entry in each session at first and then set the threshold by the quantile in the statistics to divide the session more detailed. The experiment indicates that the model obtained the highest accuracy and recall rate when the threshold was 70% quantile.

(ii) The paper compared the long short-term memory with the hidden Markov model which are commonly used to process sequence data, and it suggested that the long short-term memory model which can provide a high accuracy rate of 95.97% with a recall rate of 96.15% has much better performance than the latter one.

(iii) The proposed model could only need access log files to check the malicious behavior of webshell without any web application source files. The results show that the LSTM-based model takes 98.5% less time than previous related approach to detect whether a website has a webshell attack in a period of time.

As a remainder, related work is briefly discussed in Section 2. A complete framework is explained in Section 3. Section 4 presents the entire experiment and evaluation process in detail. At last, the conclusion is in Section 5.

2. Related Work

2.1. Outline of Webshell. Webshell is a kind of software which usually assists the administrator to manipulate the server. But in some cases, attackers will use some malicious webshells to control the server to achieve malicious purposes. For attackers, webshell is a kind of backdoor, generally written in some scripting language like ASP, PHP, or JSP. These kinds of web scripts are able to create dynamic interactive sites, so the attacker is able to control the web server through web pages. These behaviors will be recorded in the access log [10].

2.1.1. Principle of Webshell. Webshell can work by sending HTTP requests to the specific page and use some functions to execute the instruction sent by the attacker. The results of these instructions will be sent to the attacker as the response of these HTTP requests [11].

2.1.2. Classification of Webshell. According to the function and size of scripting language, webshell can be roughly divided into three categories as follows:

(i) *Big Trojan.* It has a large size and comprehensive functions for command execution, database operations, and other malicious intentions. Besides, a friendly graphical interface is applied to the big Trojan.

(ii) *One Word Trojan*. It is a Trojan with only one line of code. Due to its shortness, it is often embedded in normal files or pictures. It can perform many functions like the big Trojan when connected to the initial attack tools such as Chinese Chopper.

(iii) *Small Trojan*. It is small and easy to hide but generally only has an upload function. Since most websites have size restrictions when uploading files, attackers generally first obtain upload permission through the small Trojan and then upload the big Trojan to the website to perform key functions.

2.1.3. Escape of Webshell. In order not to be discovered by the administrator of the website, webshells usually undergo a lot of distortion. Some escape methods are as follows:

(i) Pass parameters with less common fields: while websites generally use request field to pass parameters, this method uses some unusual fields such as HTTP referrer and user agent.

(ii) Encrypt sensitive features: attackers use some common encryption algorithms such as base64 [12] and rot 13 to encrypt some key functions. For a greater probability of escape, some tools even customize encryption algorithms.

(iii) Multiple encoding and compression: hackers change the original static features of the code by multiple encoding combined with compression technology to reduce the possibility of detection.

2.2. Method of Webshell Detection. We did some research on the previous detection of webshell. At the earliest, webshell detection takes a manual identification method. This is the oldest and most traditional way to detect webshells, which places high demands on the administrators of the website. Administrators are supposed to have a comprehensive grasp of the website files and have a high recognition ability for some newly added exception files [13], such as some naming files, passby.php, pass.asp, and a.jsp. Besides, these small files should be treated carefully because there are probably one word Trojans. After finding suspicious files, we need to analyze the contents of the file. The most thorough way is to take a look at the entire file carefully, but it will take a bunch of time. A better way is to search for some sensitive functions such as *exec*(), *shell_exec*(), and *system*() and check their parameters carefully [14].

Static feature webshell detection is the hot trend of research. It is an upgraded version of manual identification, but they are almost identical. This method focuses on the features of file contents. Due to numerous features, it often uses machine learning to improve effectiveness and accuracy. For example, in [15], an approach based on optimal

thresholds was proposed to identify files containing malicious codes from web applications. The detection system will scan and find malicious codes in each file of the web application, analyzing the features including keywords, file permissions, and owner. Instead of the source file codes, Tian et al. [16] divided the POST contents in the HTTP request into several words, which were represented in the form of vectors using the Word2vec model [17] and then were input into the CNN in a fixed-size matrix. Experiments have shown that such a method can achieve high accuracy, which was also the first time that CNN [18] applied to webshell detection. However, if we just detect by signature matching, we can only do well in detecting webshells that have known features, and it does not apply to detect unknown webshells. In the literature [19], an approach that can be used to predict unknown webshells was proposed. Supervised machine learning and matrix decomposition were used to generate original and unknown webshell features by analyzing different features of known pages. The paper [20] focuses on the detection of PHP webshell, which used the text classifier fastText [21] developed by Facebook and the random forest algorithm to build the model. As a compiled intermediate language of PHP scripts, PHP opcode sequence was regarded as an important feature of webshell detection. This paper can be a good inspiration because it starts to break away from the webshell file itself.

There are also some detection methods based on other features. For example, dynamic feature detection uses the system commands, network traffic, and state exceptions used by webshell to determine the threat level of the action. Webshell is usually confused and encrypted to avoid detection of static features. When the webshell is running, system commands must be sent to the system to reach the purpose of operating the database or even the system. This method monitors and even intercepts system commands by detecting system calls and deeply detects the security of the script from the behavior mode. Zhang et al. [22] proposed a character-level content feature transformation method, which combined the features of CNN and LSTM to construct a new webshell traffic detection model. This model preserves the sequential features in network traffic and reduces the feature dimension. Experiments have shown that this model can be used to detect unknown webshells while running on large websites. Besides, Shi et al. [23] proposed a log-based method for webshell detection, which uses features such as text features, statistical features, and page association about the degree of graph theory. But the session simply uses the IP field and the user-agent field for a rough distinction, which treats all access by a user as a session. The request field is used for machine learning, and it is a text processing problem like the static detection we mentioned earlier in essence.

Among the methods we mentioned above, the defenders need to scan and look for malicious codes inside every file of the web application or every POST content in the web logs [24]. The drawbacks of these methods are the heavy workload. What is worse, with the development of encryption and obfuscation techniques, webshell detection accuracy will be much lower because webshell detection tools are mostly based on signature matching. As we mentioned before, webshell mainly works by sending HTTP requests. Based on this theory, a new direction for webshell detection is proposed which focuses on using the raw sequence data without POST contents in web logs.

3. Framework

3.1. Architecture. The purpose of this paper is to detect webshell according to the sessions extracted from web server logs without POST contents or source file codes. It is composed of several components, as illustrated in Figure 1. Firstly, we use normal log files on a large website and collect honeypot logs with webshell communication for experimental data collection. In the second part, instead of extracting the fields directly from the web logs, we use the IP field and the user-agent field to roughly divide the collected logs into different sessions, count the time interval in every session, and set the threshold to identify the session in more detail first. Then, we use the hidden Markov model and long short-term memory to build our model, respectively, and compare which one has better performance.

3.2. Raw Data Collection. In the process of generating data, we use different kinds of webshells including big Trojans, small Trojans, and one word Trojans, which differ in size and function. Besides, these webshells are placed in different depths of the website we built, and different webshell tools such as Chinese Chopper and WeBaCoo are used to initialize the attack. In this paper, the log format of the website we use is Apache, but our framework can be applied to other common server log formats such as Nginx.

3.3. Data Processing

3.3.1. Data Extraction. All the logs we collected are in the common Apache2 format. There are close to 10 fields in this format, but there is no need to use all of them. Therefore, some fields were extracted from the web logs. The whole process is shown in Figure 2. Feature sequence consists of source IP address, timestamps, user agent, status code, bytes, referrer, and request, in which the request field is subdivided into method field and path field.

The IP field and user-agent field are used to identify sessions. Because there may be multiple visitors under the same IP, it is necessary to combine user agent to make judgments. At the same time, we roughly regard a visitor as a session for the time being. The status code field and byte field can be utilized to construct feature vectors without any changes. In the method field, the GET method is represented by 1, POST method is represented by 2, and other methods are represented by 0. Meanwhile, in the referrer field, "–" is represented by 0, while 1 for the website, 2 for the web crawler, and 3 for the external websites. In the timestamps field, we calculate the time difference between entries in every session and fill 0 in the last vector of every session.

The path field is encoded with the degree of relevance of each entry access path in the session. The first entry of each session is coded as −1 because there is no access in front of it,

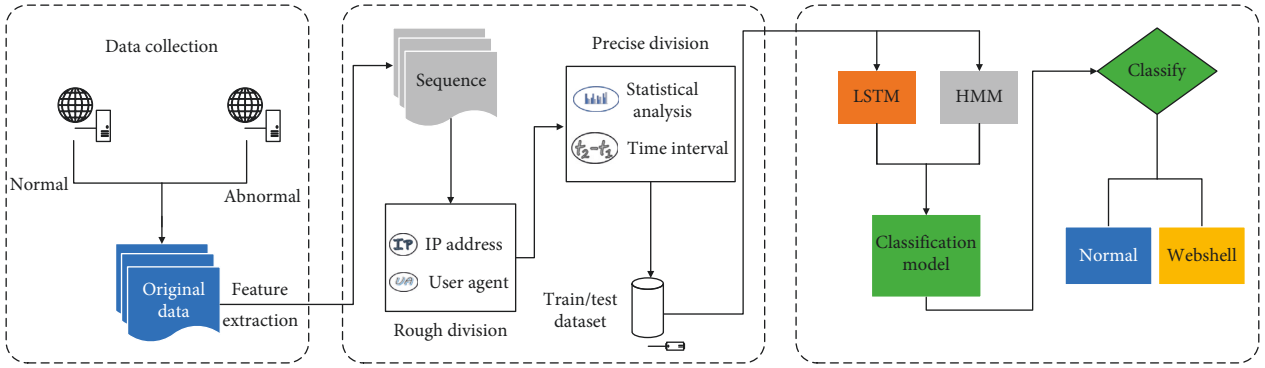


Figure 1: The architecture of the proposed method.

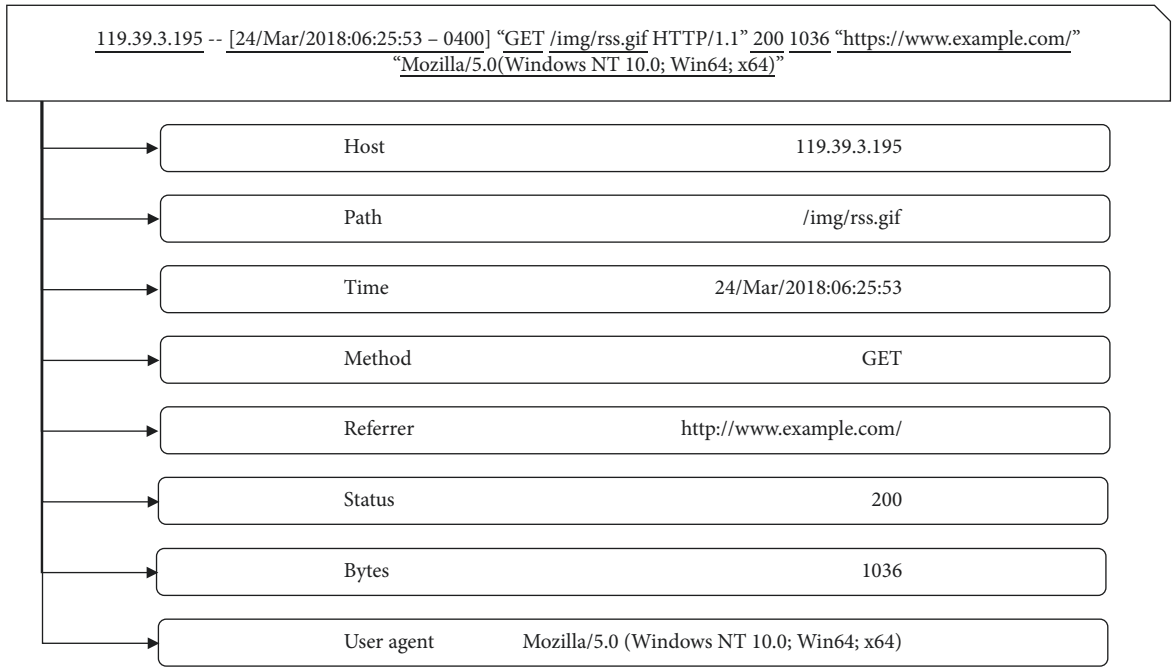


Figure 2: Feature extraction based on log entry.

and the relative relationship cannot be discriminated; from the second file, the access to the same file is marked as 0, and when accessing different files, the distance between different files is calculated by the number of directory switches plus one to encode. The process of encoding all paths for a session is shown in Figure 3. When all the feature fields are processed, a session will transform into a sequence of features as illustrated in Table 1. Finally, the feature vector we get will be six-dimensional including byte field, method field, path field, referrer field, status code field, and time interval.

3.3.2. Session Identification. In the previous section, we just made a rough distinction between the sessions in every log file, treating a visitor as a session. But more often, a visitor can access at different times and generate multiple sessions. Thus, we use a more scientific statistical method for session identification.

We calculate and count the time interval between entries in every session, which is roughly generated in the previous section. After counting the time intervals in all sessions, we sort all of them and explore the most appropriate quantile as a threshold. We compare every time interval to a threshold, and if the time interval is greater than the threshold, the session will be subdivided into two smaller sessions, and so forth. The whole process is illustrated in Figure 4.

When all the data have been processed, the framework will check all the sessions and delete those sessions that contain only one entry, as it assumes that if a user has only one access, it is highly unlikely that there is a webshell communication.

3.4. Classification Model. Webshell detection is a two-class task, and because of variable length serialization in the log sequence, we choose long short-term memory and the hidden Markov model to construct the classification model, respectively.

3.4.1. Long Short-Term Memory. Long short-term memory networks are well suited to classify and process based on

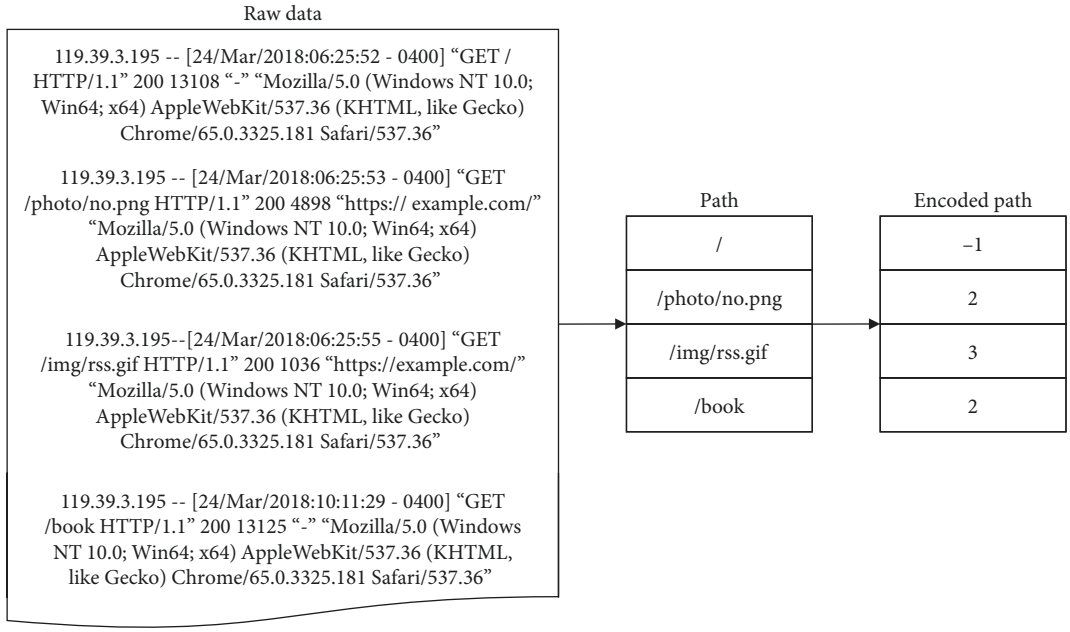


Figure 3: The process of encoding all paths for a session.

Table 1: Session transformation.

Log messages	Feature vector
Log entry0	$[bytes_0, method_0, path_0, referrer_0, statuscode_0, t_1 - t_0]$
Log entry1	$[bytes_1, method_1, path_1, referrer_1, statuscode_1, t_2 - t_1]$
Log entry2	$[bytes_2, method_2, path_2, referrer_2, statuscode_2, t_3 - t_2]$
⋮	⋮
Log entry$n$	$[bytes_n, method_n, path_n, referrer_n, statuscode_n, 0]$

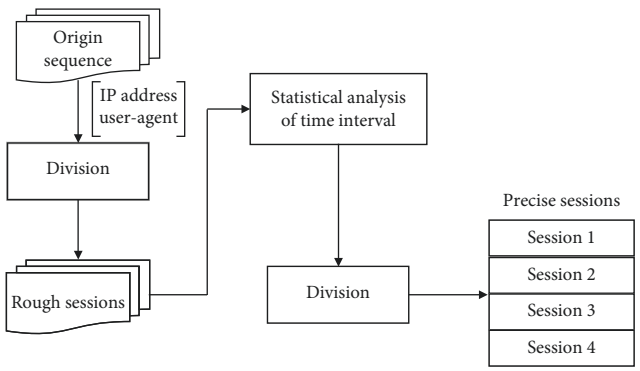


Figure 4: Schematic diagram of session identification.

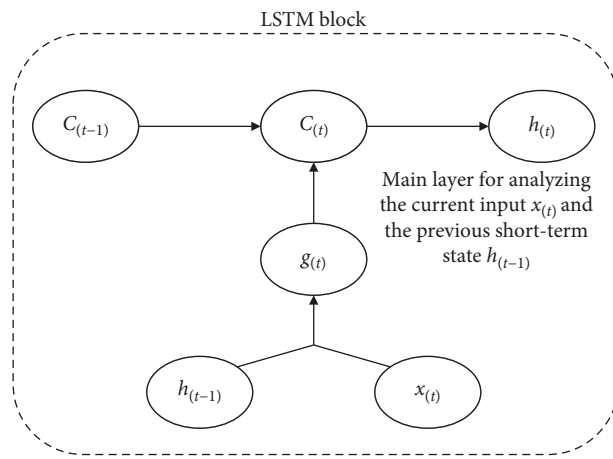


Figure 5: The core of LSTM.

sequence data with its faster convergence and the ability to detect long-term dependencies in data. The concept was originally proposed in 1997 by Hochreiter and Schmidhuber [25]. The emergence of LSTM is mainly to solve the problem of gradient disappearance and gradient explosion in long sequence training. Compared with traditional RNN where there is only one delivery state, the LSTM has two delivery states, namely, the long-term state $c_{(t)}$ and the short-term state $h_{(t)}$. Besides, it introduces three "gates" to control the long-term state, as illustrated in Figure 5.

(i) Forget gate $f_{(t)}$: control which of the long-term states of last moment $c_{(t-1)}$ should be discarded

(ii) Input gate $i_{(t)}$: control which parts of the current time input $x_{(t)}$ and the last short-term state $h_{(t-1)}$ will be added to the long-term state

(iii) Output gate $o_{(t)}$: control which parts of the current long-term state $c_{(t)}$ should be output as $h_{(t)}$

The model which called SB-LSTM (session-based long short-term memory) consists of four LSTM layers, each of which has sixteen neurons, and a dense layer. The LSTM layer will learn the feature of variable length sequences which are activated by tanh function and logistic function. The dense layer acts as a "classifier" throughout the LSTM, which can map the learned "distributed feature representation" to the sample tag space. At last, the detection model used categorical cross entropy as the loss function, and the optimizer is Adam.

In more detail, the input data size is $m \times n \times 6$, m is the number of sessions, and n is the maximum length of all sessions. Current long-term state $c_{(t)}$ and output $y_{(t)}$ are computed as follows:

$$\begin{aligned} g_{(t)} &= \tanh\left(W_{xg}^{T} \cdot x_{(t)} + W_{hg}^{T} \cdot h_{(t-1)} + b_g\right), \\ c_{(t)} &= f_{(t)} \otimes c_{(t-1)} + i_{(t)} \otimes g_{(t)}, \\ y_{(t)} &= h_{(t)} = o_{(t)} \otimes \tanh\left(c_{(t)}\right), \end{aligned} \tag{1}$$

where W_{xg} denotes the weight matrix of the main layer connected to the input vector $x_{(t)}$ of size 1×6, while W_{hg} denotes the weight matrix of the main layer connected to the previous short-term state $h_{(t-1)}$. b_g represents the coefficient of variation for the main layer.

The architecture of this model for a session is depicted in Figure 6.

3.4.2. Hidden Markov Model. The hidden Markov model (HMM) is a statistical Markov model, a highly effective means of modeling a family of unaligned sequences [26, 27], which describes a hidden Markov chain stochastically generating unobservable state sequences and then generating a stochastical observation sequence from each state. As its most remarkable features, the state at any time t depends only on the state of the previous moment, while it is independent of the time t, the state, and the observation at other times. Besides, it also assumes that observations at any time depend only on the state at that moment, independent of other observations and states.

In this paper, we mainly focus on supervised learning method of the hidden Markov chain. The train data include observation sequences O and corresponding state sequences I, which can be written as

$$\{(O_1, I_1), (O_2, I_2), \ldots, (O_s, I_s)\}. \tag{2}$$

We can use the maximum likelihood estimation method to estimate the parameters of the hidden Markov model:

(1) We assume that the sample is in the state i at time t and the frequency of transition to state j at time $t + i$ is A_{ij}, and then the probability estimate of the transition state is computed as follows:

$$\widehat{a_{ij}} = \frac{A_{ij}}{\sum_{j=1}^{N} A_{ij}}, \quad i = 1, 2, \ldots, N;\ j = 1, 2, \ldots, N. \tag{3}$$

(2) We assume that the sample state is j and the frequency of observation k is B_{jk}, and then the probability of observation k when the state is j can be written as

$$\widehat{b_{jk}} = \frac{B_{jk}}{\sum_{k=1}^{M} B_{jk}}, \quad j = 1, 2, \ldots, N;\ k = 1, 2, \ldots, M. \tag{4}$$

(3) The initial state probability π is estimated as the frequency of the initial state of i in the S samples. The architecture of this model is illustrated in Figure 7.

In more detail, the input data size is $x \times 6$ and x is the number of entries in all sessions.

4. Experiment and Evaluation

In this section, we will introduce the detailed composition of the dataset, the experiment details, and the results.

4.1. Dataset. There are two datasets for our experiments, one for the model training and the other for the model test. As for the first dataset, a honeypot [28, 29] was built to capture and analyze webshell attacks. Tens of testers were invited to attack this honeypot. One word Trojans, small Trojans, and big Trojans which were coded in different program languages were provided to these testers. We collected a total of 13,986 log entries from the honeypot as negative samples, while 57,160 logs entries were collected from real-world websites as positive samples. In the process of training, we also used a 10-folder cross validation to perform a simple verification of the model. As for the latter one, a real-world website which contains a total of 2324 files and 248 k lines of log entries in total was utilized to test. Since the log entries with webshell communication in the real environment may only account for a few percents or even a few thousandths in the log file, we control the positive and negative sample ratio of the dataset to be around 6 : 1 in experiments. After feature vector extraction and session identification, all log entries become independent sessions. Every session consists of six sequences, which are the byte sequence of the HTTP response, the method sequence, the path sequence that is encoded by the degree of association, the referrer sequence that is divided into different cases for encoding, the status code sequence, and the time interval sequence. Figure 8 is an overview of our model.

4.2. Experiment Design. To evaluate the performance of our model, we first explored the effects of different quantiles as thresholds and selected the best values for next experiments. Then, we validated the effectiveness of the session identification method based on the time interval and quantile. In addition, we compare long short-term memory with the hidden Markov models, which are commonly used in sequence data processing, and find the one with better performance. Finally, we leverage a real-world website to test the SB-LSTM model and compare it with results from other research.

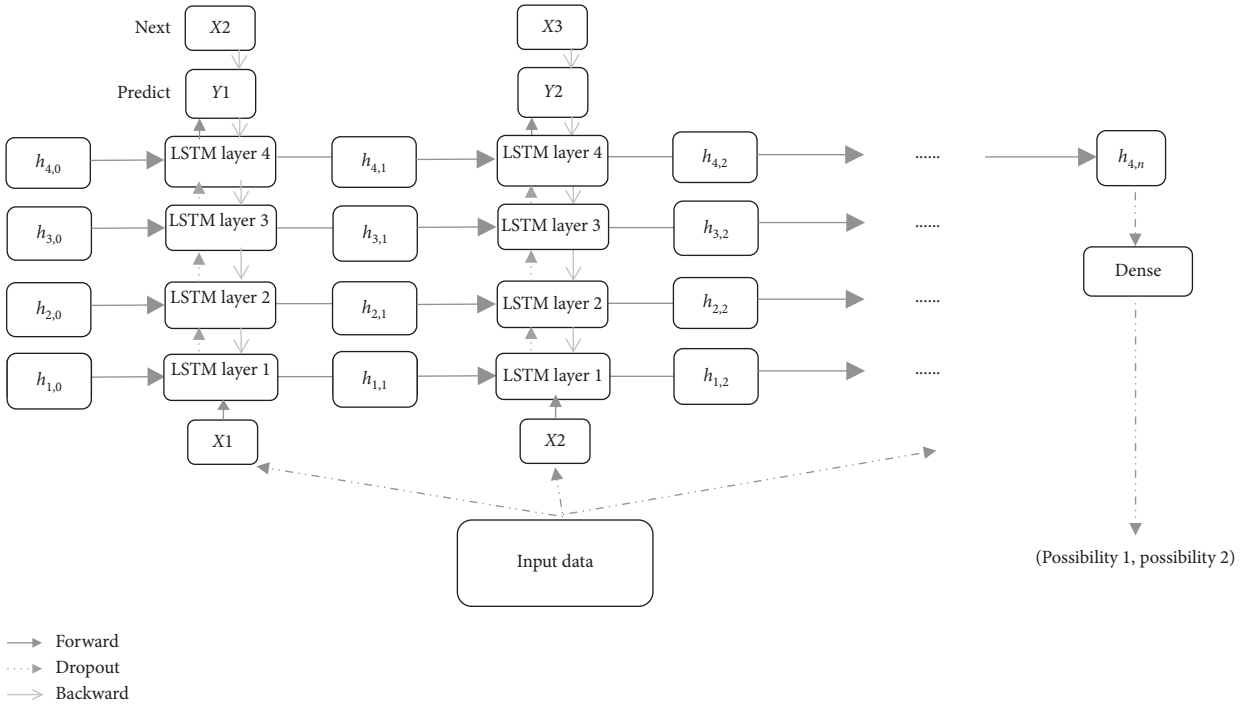


Figure 6: The architecture of LSTM.

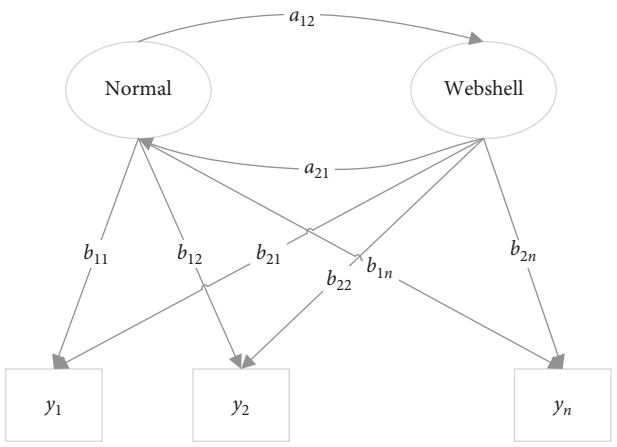


Figure 7: The architecture of HMM.

All the experiments were performed in a PC machine with an i7-7700HQ processor, 16 GB of memory, and a GeForce GTX 1060 GPU which has 6 GB of memory. The SB-LSTM model is implemented by Tensorflow [30], and the HHM is implemented by seqlearn [31].

Before using the raw data to build the model, we did a standardization to make all the data appropriate for the neural network. Data were processed according to the following steps:

(i) Since we need to input a 3-dimensional matrix when constructing the SB-LSTM model, we chose Max-Min scaling normalization to reduce the effect of padding 0 when normalizing. After scaling, all data are between 0 and 1. The function of doing Max-Min scaling normalization is

$$x^* = \frac{x - \min}{\max - \min}. \tag{5}$$

(ii) The label of all the training data is stored in a list, 0 for normal access and 1 for webshell access.

(iii) When we used long short-term memory to build our model, we must make all data to become a matrix, so we padded the length of the sequence to z, which is the maximum session length. Besides, we held a list to show all the valid length (original length) of the data to ensure the number of iterations in LSTM.

(iv) When we used the hidden Markov model to build our model, we only need to put all the vectors into a sequence because of the input data size of the hidden Markov model.

We use 10-folder cross validation [32]. The dataset is equally divided into 10 subsets, each of which is tested once and the rest as a training set. The cross validation is repeated 10 times, one subset is selected each time as a test set, and the average cross validation recognition accuracy rate of 10 times is taken as a result. The dataset used for the experiment is unbalanced, so we choose accuracy, recall, precision, $F1$ score [33], the receiver operating characteristics (ROC) curves [34], and area under curve (AUC) measure for evaluating the proposed method. Besides, we can visualize the relation between TPR and FPR of a classifier. These indicators can be expressed as

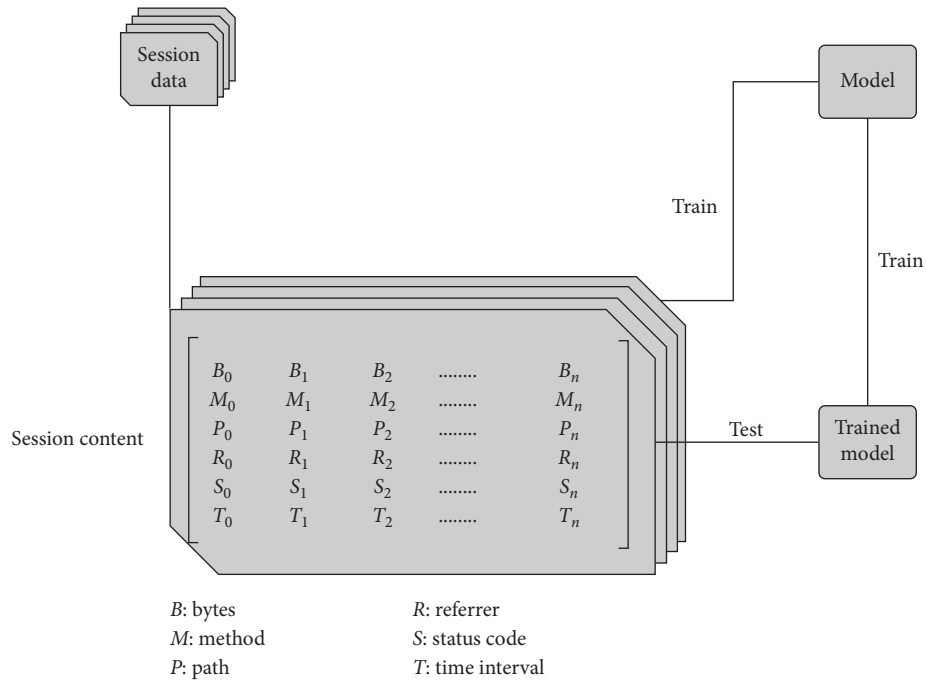


Figure 8: Overview of webshell detection with our model.

$$\begin{aligned}
\text{Accuracy} &= \frac{1}{|C|}\sum_{i=1}^{|C|}\frac{\text{TP}_i + \text{TN}_i}{\text{TP}_i + \text{TN}_i + \text{FP}_i + \text{FN}_i},\\
\text{Recall} &= \frac{1}{|C|}\sum_{i=1}^{|C|}\frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i},\\
\text{Precision} &= \frac{1}{|C|}\sum_{i=1}^{|C|}\frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i},\\
F1\ \text{score} &= 2\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}},\\
\text{AUC} &= \frac{\sum_{i\in\text{positiveclass}}\text{rank}_i - (M(1+M)/2)}{M \times N}.
\end{aligned} \tag{6}$$

where M is the number of positive samples and N is the number of negative samples. The score indicates the probability that each test sample belongs to a positive sample, while *rank* is a positive sample set sorted in the descending order based on score.

4.3. Experiment Results. First, we explored the influence of different thresholds in session identification. The thresholds were set to 55%, 60%, 65%, 70%, 75%, 80%, 85%, and 90% quantile, respectively. The comparison results are shown in Table 2.

As is shown in Table 2, performance is improved with an increasing threshold in a certain range and reaches the highest value when the threshold is 70%. Moreover, when the performance reaches the highest, it will continuously decrease with a rising threshold. Excessive thresholds can decrease the performance of the model because two accesses that were not originally part of the same session are placed in the same session. In comparison with the threshold of 55% to 90%, we find the appropriate threshold is 70%, and we use it for the next experiments.

Table 2: The influence of different thresholds in session identification.

Threshold (%)	Accuracy	Recall	Precision	$F1$ score
55	0.932	0.8624	0.8757	0.8690
60	0.9364	0.869	0.8958	0.8822
65	0.9598	0.9568	0.8977	0.9263
70	0.9597	0.9615	0.9036	0.9317
75	0.9696	0.9368	0.9009	0.9185
80	0.9376	0.9401	0.8518	0.8937
85	0.8974	0.7228	0.6932	0.7077
90	0.8779	0.7186	0.6418	0.6780

Then, we validated the effectiveness of session identification based on the time interval and the quantile. We counted the number of access sessions for every user, where the user is identified by the IP field and the user-agent field. The top 100 users with the largest number of sessions are shown in Figure 9.

As is illustrated in Figure 9, most users have multiple sessions, up to a maximum of 166. Therefore, it is quite necessary for us to conduct a more detailed session identification.

In addition, we compare the performance of two machine learning models. The comparisons are presented in Table 3.

As is shown in Table 3, SB-LSTM has a much better performance than the HHM-based model. The recall rate of the HMM-based model under such dataset training is close to 0. The reason that the HMM assumes that the state at any

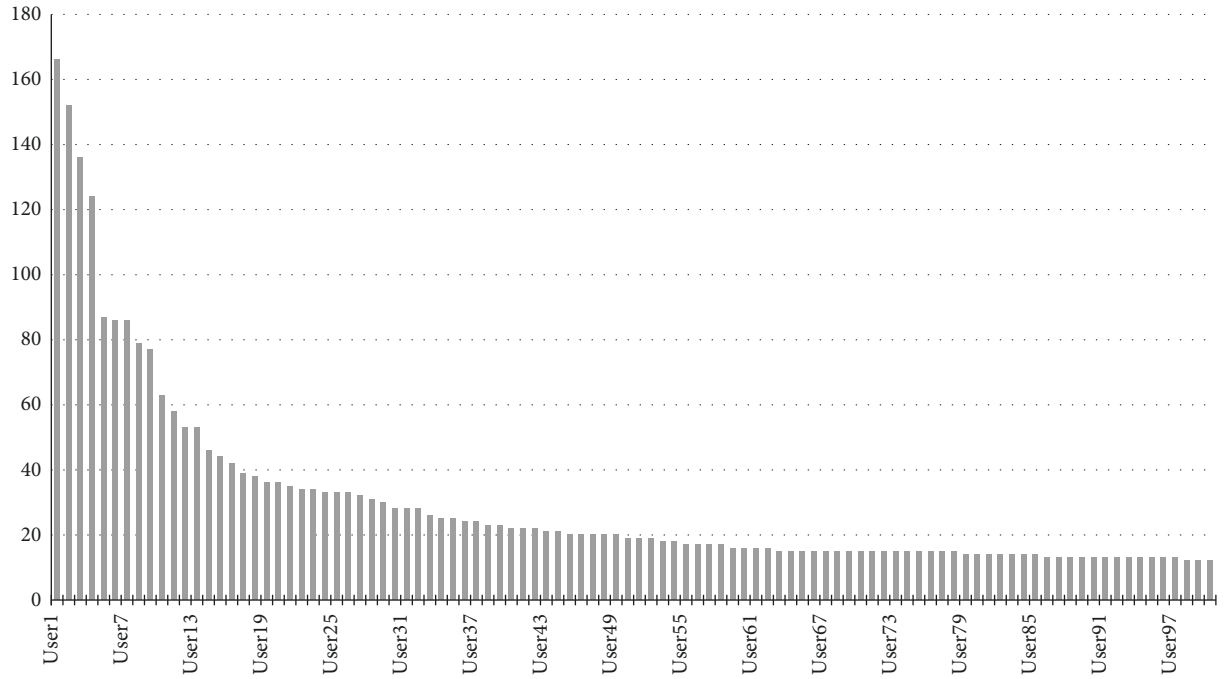


Figure 9: The top 100 users with the largest number of sessions.

Table 3: Comparison of two machine learning methods.

Model name	Accuracy (%)	Recall (%)
SB-LSTM	95.97	96.15
HMM-based model [31]	68.27	0.00

time depends only on the state of the previous moment might lead to the low recall. Besides, the HMM-based model also has a much lower accuracy than SB-LSTM. The ROC is shown in Figure 10. The curve shows the SB-LSTM model could achieve an effective and accurate result in which the true positive rate could reach 0.8 when the false positive rate only reaches 0.01.

Afterwards, we found the webshell communication, and we can easily find the webshell by counting all the access paths of the session. Besides, if the administrator is familiar with the files included in website dictionary, he may not need to perform statistics to find the webshell.

At last, we leverage a real-world website to test the SB-LSTM model and compare it with results from other research. We first explored the influence of the different sizes of logs on the model's runtime. The experiment used two log files including a recent one and the largest one the website can provide. Besides, we ran this experiment three times and got the average as results. The results are presented in Table 4. Then, we compared these to the result of the cited related work. We downloaded all the source files of the website and reproduced a cited related work, which is a file-based detection method. For the purpose of maximizing the experimental results, we used the largest log the website can provide which records all access to the website for nearly a year and a half to compare. We do the same things in terms of the final results, and the comparison results are shown in Table 5.

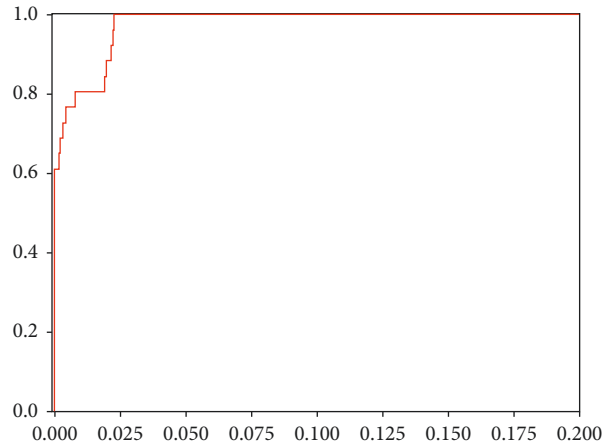


Figure 10: ROC curve based on SB-LSTM.

Table 4: The influence of the different sizes of logs.

Model name	Sizes	Entries	Time
SB-LSTM	177 kB	908	0.5897 s
	46.6 MB	248433	28.7127 s

Table 5: Comparison of different detection methods.

Model name	Sizes/amount	Time
SB-LSTM	46.6 MB	28.7127 s
FRF-WD [20]	2324	39.8415 s

As is illustrated in Table 4, when the size of the log is quite small, SB-LSTM can get the result of detection very quickly, which is efficient for detecting whether there is webshell communication for a certain period of time.

As revealed in Table 5, even if we use all the log entries since the website was built, the runtime required for analyzing is 27.9% less than the result of the cited related work.

Compared with the results of the first experiment, it demonstrated that the SB-LSTM model is efficient for detecting whether there is webshell communication for a certain period of time, as it requires to scan all the source files in the common approach while the only input for the SB-LSTM model is the recent log.

5. Conclusion

In the previous studies, a huge number of features were extracted from webshell, and these features were regarded as keywords to judge whether there is a webshell or not. However, it is almost certain that these approaches would be less useful with the gradual development of encryption and confusion technology. This paper mainly focuses on the challenge that detects webshell out of itself. Instead of leveraging POST contents, source file codes, or receiving traffic, the framework we proposed uses sessions generated from the website's logs, which highly reduces the cost of time and space but maintains a high recall rate and accuracy. Features were extracted in raw sequence data in the web logs, and a statistical method was applied to identify sessions precisely. The results of experiments show that 70% quantile can be the right threshold that makes the model obtain the highest accuracy and recall rate, and the long short-term memory which can achieve a high accuracy rate of 95.97% with a recall rate of 96.15% has much better performance than the hidden Markov model on webshell detection. Moreover, the experiment demonstrated the high efficiency of the proposed approach in terms of the runtime, as it takes 98.5% less time than the cited related approach to get the results. In order to be closer to the real-world application, the model can be employed to identify webshell files after the webshell communication is detected by using a statistical method.

Data Availability

The Web logs data used to support the findings of this study have not been made available because it was extracted from real websites, and contains many sensitive information.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the Fundamental Research Funds for the Central Universities and the Sichuan University Postdoc Research Foundation under Grant 19XJ0002.

References

[1] CNCERT weekly report," 2019, http://www.cert.org.cn/publish/english/upload/File/Weekly%20Report%20of%20CNCERT-Issue%207%202019.pdf.

[2] D. H. Tony Lee and I. Ahl, "Breaking down the China chopper web shell-part I," 2019, https://www.fireeye.com/blog/threat-research/2013/08/breaking-down-the-china-chopper-web-shell-part-i.html.

[3] B. Yong, X. Liu, Y. Liu, H. Yin, L. Huang, and Q. Zhou, "Web behavior detection based on deep neural network," in *Proceedings of the IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing*, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), pp. 1911–1916, Hong Kong, China, July 2018.

[4] L. Y. Deng, D. L. Lee, Y.-H. Chen, and L. X. Yann, "Lexical analysis for the webshell attacks," in *Proceedings of the 2016 International Symposium on Computer, Consumer and Control (IS3C)*, pp. 579–582, Xi'an, China, July 2016.

[5] V.-G. Le, H.-T. Nguyen, D.-N. Lu, and N.-H. Nguyen, "A solution for automatically malicious web shell and web application vulnerability detection," in *Proceedings of the International Conference on Computational Collective Intelligence*, pp. 367–378, Halkidiki, Greece, September 2016.

[6] J. Wang, Z. Zhou, and J. Chen, "Evaluating CNN and LSTM for web attack detection," in *Proceedings of the 2018 10th International Conference on Machine Learning and Computing*, pp. 283–287, Macau, China, February 2018.

[7] W. Yang, B. Sun, and B. Cui, "A webshell detection technology based on http traffic analysis," *Innovative Mobile and Internet Services in Ubiquitous Computing*, in *Proceedings of the International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 336–342, Matsue, Japan, July 2018.

[8] J. Kim, D.-H. Yoo, H. Jang, and K. Jeong, "Webshark 1.0: a benchmark collection for malicious web shell detection," *JIPS*, vol. 11, no. 2, pp. 229–238, 2015.

[9] P. M. Wrench and B. V. Irwin, "Towards a php webshell taxonomy using deobfuscation-assisted similarity analysis," in *Proceedings of the 2015 Information Security for South Africa (ISSA)*, pp. 1–8, Johannesburg, South Africa, July 2015.

[10] Z. Meng, R. Mei, T. Zhang, and W.-P. Wen, "Research of linux webshell detection based on SVM classifier," *Netinfo Security*, vol. 5, pp. 5–9, 2014.

[11] O. Starov, J. Dahse, S. S. Ahmad, T. Holz, and N. Nikiforakis, "No honor among thieves: a large-scale analysis of malicious web shells," in *Proceedings of the 25th International Conference on World Wide Web*, pp. 1021–1032, Montreal, Canada, April 2016.

[12] S. Josefsson, "The base16, base32, and base64 data encodings," 2009, http://www.hjp.at/doc/rfc/rfc3548.html.

[13] Z.-H. Lv, H.-B. Yan, and R. Mei, "Automatic and accurate detection of webshell based on convolutional neural network," in *Proceedings of the China Cyber Security Annual Conference*, pp. 73–85, Beijing, China, August 2018.

[14] Compromised web servers and web shells-threat awareness and guidance," 2017, https://www.us-cert.gov/ncas/alerts/TA15-314A.

[15] T. D. Tu, C. Guang, G. Xiaojun, and P. Wubin, "Webshell detection techniques in web applications," in *Proceedings of the Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pp. 1–7, Hefei, China, 2014.

[16] Y. Tian, J. Wang, Z. Zhou, and S. Zhou, "CNN-webshell: malicious web shell detection with convolutional neural network," in *Proceedings of the 2017 VI International*

Conference on Network, Communication and Computing, pp. 75–79, Kunming, China, December 2017.

[17] T. Walkowiak, S. Datko, and H. Maciejewski, "Bag-of-words, bag-of-topics and word-to-vec based subject classification of text documents in polish-a comparative study," *Contemporary Complex Systems and Their Dependability*, pp. 526–535, 2018.

[18] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *The Journal of Physiology*, vol. 195, no. 1, pp. 215–243, 1968.

[19] X. Sun, X. Lu, and H. Dai, "A matrix decomposition based webshell detection method," in *Proceedings of the 2017 International Conference on Cryptography, Security and Privacy*, pp. 66–70, Wuhan, China, March 2017.

[20] Y. Fang, Y. Qiu, L. Liu, and C. Huang, "Detecting webshell based on random forest with fasttext," in *Proceedings of the 2018 International Conference on Computing and Artificial Intelligence*, pp. 52–56, Chengdu, China, March 2018.

[21] Package Information [eb/ol], 2019, https://pecl.php.net/package/vld.

[22] H. Zhang, H. Guan, H. Yan et al., "Webshell traffic detection with character-level features based on deep learning," *IEEE Access*, vol. 6, pp. 75268–75277, 2018.

[23] L. Shi and Y. Fang, "Webshell detection method research based on web log," *Journal of Information Security Research*, vol. 1, p. 11, 2016.

[24] R. Sasi, "Web backdoors-attack, evasion and detection," in *Proceedings of the C0C0N Sec Conference*, pp. 989–1003, Cochin, India, 2011.

[25] J. Schmidhuber and S. Hochreiter, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[26] R. Hughey and A. Krogh, "Hidden markov models for sequence analysis: extension and analysis of the basic method," *Bioinformatics*, vol. 12, no. 2, pp. 95–107, 1996.

[27] B. Schuller, G. Rigoll, and M. Lang, "Hidden markov model-based speech emotion recognition," in *Proceedings of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Atlanta, GA, USA, April 2003.

[28] I. Kuwatly, M. Sraj, Z. Al Masri, and H. Artail, "A dynamic honeypot design for intrusion detection," in *Proceedings of the IEEE/ACS International Conference on Pervasive Services*, Lebanon, July 2004.

[29] C. Kreibich and J. Crowcroft, "Honeycomb," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 1, pp. 51–56, 2004.

[30] TensorFlow," 2019, https://www.tensorflow.org/.

[31] L. larsmans, "seqlearn," 2016, https://github.com/larsmans/seqlearn.

[32] R. Kohavi et al., "A study of cross-validation and bootstrap for accuracy estimation and model selection," *IJCAI*, vol. 14, no. 2, pp. 1137–1145, 1995.

[33] C. Goutte and E. Gaussier, "A probabilistic interpretation of precision, recall and *f*-score, with implication for evaluation, Lecture Notes in Computer Science," in *Proceedings of the European Conference on Information Retrieval*, pp. 345–359, Santiago de Compostela, Spain, March 2005.

[34] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

Hindawi
Security and Communication Networks
Volume 2019, Article ID 2317976, 12 pages
https://doi.org/10.1155/2019/2317976

Research Article

Evaluation of Deep Learning Methods Efficiency for Malicious and Benign System Calls Classification on the AWSCTD

Dainius Čeponis and Nikolaj Goranin

Department of Information Systems, Vilnius Gediminas Technical University, Saulėtekio al. 11, LT-10223, Vilnius, Lithuania

Correspondence should be addressed to Dainius Čeponis; dainius.ceponis@vgtu.lt

Received 1 April 2019; Revised 24 July 2019; Accepted 16 August 2019; Published 11 November 2019

Guest Editor: Hyoungshick Kim

Copyright © 2019 Dainius Čeponis and Nikolaj Goranin. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The increasing amount of malware and cyberattacks on a host level increases the need for a reliable anomaly-based host IDS (HIDS) that would be able to deal with zero-day attacks and would ensure low false alarm rate (FAR), which is critical for the detection of such activity. Deep learning methods such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are considered to be highly suitable for solving data-driven security solutions. Therefore, it is necessary to perform the comparative analysis of such methods in order to evaluate their efficiency in attack classification as well as their ability to distinguish malicious and benign activity. In this article, we present the results achieved with the AWSCTD (attack-caused Windows OS system calls traces dataset), which can be considered as the most exhaustive set of host-level anomalies at the moment, including 112.56 million system calls from 12110 executable malware samples and 3145 benign software samples with 16.3 million system calls. The best results were obtained with CNNs with up to 90.0% accuracy for family classification and 95.0% accuracy for malicious/benign determination. RNNs demonstrated slightly inferior results. Furthermore, CNN tuning via an increase in the number of layers should make them practically applicable for host-level anomaly detection.

1. Introduction

The best method to detect unsanctioned or malicious usage of a company's system is to use an intrusion detection system (IDS). IDSs are classified into two main types: network-based IDS (NIDS) and host-based IDS (HIDS) [1]. NIDSs work on the network level and are capable of detecting any malicious activity that can be observed on a company's local network. The HIDSs monitor the activities on end-user machines. They can collect and analyze information such as machine parameters (CPU and RAM usage), modified files, modified registry items (Windows operating system), system calls, etc. While research on NIDS has reached a relatively advanced level and a number of anomaly-based solutions are available on the market [2], HIDSs are stuck in signature-based or file-integrity monitoring approach, making them immune to zero-day attacks [3]. The importance of HIDS becomes critical [4, 5] and requires development of anomaly-based solutions. The earlier approaches based on threshold parameters were not successful because of high false alarm rate (FAR), but recent advances in deep learning (DL) techniques demonstrate the potential of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) in activity classification. Therefore, comparative analysis of such methods is necessary to evaluate their efficiency in malicious activity classification and ability to distinguish malicious and benign activity [6]. The evaluation of these methods allows the selection of the most appropriate and accurate method for anomaly-based HIDS development. Accuracy plays a crucial role because the high false-positives rate would lead to distrust in the system and ignorance of alerts.

Many researchers use machine learning (ML) methods to achieve proper IDS accuracy in detecting intrusive actions. Training and testing data are required to apply ML methods. Most of the recent research was conducted with the old datasets generated in 1998-1999 [7, 8] named DARPA and KDD Cup 99, respectively. Overall, 42% and 20% of the researchers used DARPA dataset and KDD Cup 99, respectively [9]. Both databases focused on NIDS-related

data and lacked the information required to train HIDS-suitable methods.

Some attempts [10] have been made to fulfill the growing need for Windows-oriented HIDS datasets. According to statcounter.com, Windows operating systems were still used by more than 70% of the desktop users in 2018 (see Figure 1) [11].

One of the latest HIDS-related datasets for Windows OS is ADFA-IDS [12]. It has a collection of system calls produced on Linux and Windows operating systems. However, ADFA family datasets only have minimal data required for intrusion detection, as they contain only system call identification—system dynamic link library (dll) file name and the called function name. Even the authors of the ADFA-IDS agree that the dataset is incomplete: only basic information was collected, and an insufficient number of vulnerabilities were used to generate malicious activity [13].

We have previously generated AWCSTD to fulfill the demand for a more extended dataset for Windows operating system [12]. It uses more malware samples (12110) and collects more system calls sequences (112.56 million) than any similar public dataset. Most importantly, it also contains 16.3 million systems calls generated by 3145 benign software samples at present. The same method has been used to generate benign system calls sequences as the one previously used in [12] for malignant ones. In total, six virtual machines with Windows 7 operating system preinstalled were utilized. The virtual machines had tools such as Notepad++, 7zip, and data logging tools installed. This allows using the dataset not only for training neural networks on classifying malicious activities but also for training them in distinguishing between malicious and legal activity in general.

In this paper, we present the results of efficiency evaluation for RNNs and CNNs achieved with AWSCTD with different initial data parameters. The remainder of the paper is organized as follows: the Introduction gives a general view on the importance of datasets for HIDS training and the need for evaluating the efficiency of different DL methods; the Related Work section presents results achieved by other research teams; the Materials and Methods part of the article describes the dataset preparation and methods used; the Results and Discussion present the results with comments on their reasons and applicability; Conclusions summarize the results and define future work.

1.1. Related Work. Attackers often use various techniques to transform and hide malicious activity from the signature-based IDS [14]. Anomaly-based techniques are used to tackle such problems. They have not only introduced a better detection rate of unknown attacks but also increased the number of false-positives [15] because of primitive approaches applied. Advances in deep learning methods combined with extensive training datasets are required to build a benign behavior profile and decrease the FAR.

Several ML classification and clustering methods such as neural networks, support vector machines, k-means clustering, k-nearest neighbors, and decision trees [16–18] have been used to improve anomaly-based IDS. The authors of

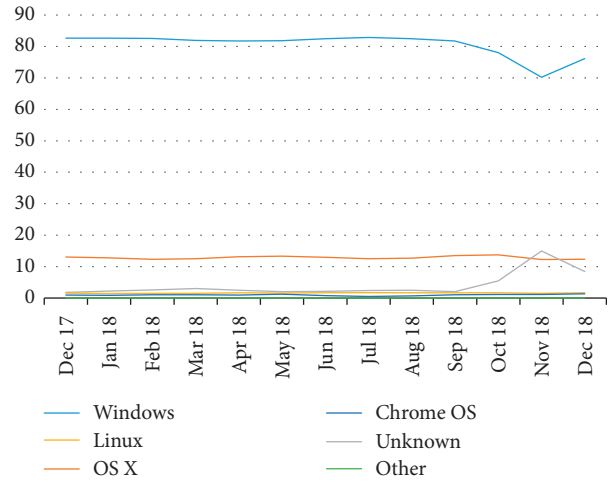


Figure 1: Desktop operating systems market share in 2018.

ADFA-WD (Windows-based) have achieved a 72% detection rate with Naïve Bayes method, and the data were based on transforming system call traces into frequency vectors [13]. Later, the authors of [18] achieved a 61.2% detection rate with ADFA-LD (Linux-based) when CNNs were used. In [19], the authors claim 86.4% accuracy in malicious activity classification with the help of the hybrid neural network, but the dataset was not published, thereby offering no chance to check the accuracy of the results. Similar classification was performed by us on AWSCTD using SVM, and an accuracy of 92.4% was achieved [17]. In addition, the tested decision trees method, which is lighter in terms of training and testing times, has shown comparable results of 92.1% accuracy. This is an essential point, as fast model training is a critical factor in cybersecurity, where new attack samples are introduced very often. Hence, the results in [13, 18] demonstrate very high FAR, whereas the results in [17, 19] do not solve the malicious/benign classification task.

Since 2006, deep-structured learning, commonly called deep learning or hierarchical learning, has emerged as a new area of machine learning research [20]. The architecture of deep neural networks is based on many layers of neural networks (NNs). Artificial NNs (ANNs) were naturally developed from biological neural networks. The first paper on neural networks was produced in 1943; professors McCulloch and Pitts published a paper titled "A Logical Calculus of the Ideas Immanent in Nervous Activity" that logically explained the human neural network and conceptualized the ANNs for the first time in history [21]. An artificial neural network consists of a group of processing elements that are interconnected and convert a set of inputs to a set of preferred outputs. The result of the transformation is determined by the characteristics of the elements and the weights associated with the interconnections among them. The network can adapt to the desired outputs by modifying the connections between the nodes [22]. A fundamental property of neural networks is the concept of programming by example. A large number of weights makes it difficult to fix them and obtain the desired result. Instead, the network is programmed by example and repetition. It is trained by

presenting input-output pairs repeatedly. Each time an input is presented, the network guesses the output. The output part of the input-output pair is used to determine whether the network is right or wrong. If wrong, the network is corrected by a learning algorithm using a gradient method on the output error to modify the weights. After each modification, the network gets closer to the desired transfer function as represented by the sample base [23].

The most significant disadvantage of applying neural networks to intrusion detection is the "black box" nature of the neural network. The "Black Box Problem" has overwhelmed neural networks in many applications [24]. This Black Box neural networks feature does not allow the researchers to clearly see and analyze learning results. This also makes the network tinkering process more difficult—researchers cannot analyze and modify networks for better outcomes. DNNs and new hardware capabilities have revived the current state of the ANN research. Two powerful DNN designs were introduced: convolutional neural network (CNN) and recurrent neural network (RNN). CNNs, or ConvNet, are mainly applied for the image recognition tasks because they can scale adequately on large images. They are based on several convolutions and pooling layers combinations that lead to the last, simple ANN layer for final classification. The usage of convolution and pooling layers allows reduction of the feature maps size [25, 26]. RNN is capable of working with time series-based data. The development of RNNs began in 1997 when long short-term memory (LSTM) networks were introduced [27]. The natural capability to accept and work with sequences has allowed to show outstanding performance in speech recognition and machine translation [28]. In 2014, the gated recurrent unit (GRU) was introduced for RNN [29]. It is similar to LSTM but has fewer parameters because it lacks an output gate. GRU is mainly applied in natural language analysis and translation.

Primary DNN applicability for anomaly detection still concentrates on NIDS and the use of KDD dataset [30–32]. The most popular method is RNN with LSTM that provides up to 96% accuracy. However, CNN has also demonstrated applicability for such tasks [19]. This encouraged us to evaluate both CNN and RNN with AWSCTD for anomaly detection (malicious/benign classification) on the end-user machine level.

2. Materials and Methods

2.1. Dataset. For our experiment, AWSCTD containing system calls sequences from Windows OS was used [12]. It was generated using publicly available malware files from Virus Share [33] and publicly available information about any malware found from Virus Total [34]. Later, the collected database was updated with the additional information provided by the Virus Total that included scan results and behavioral information.

For experiments described in this article, AWSCTD was appended with 16.3 million system calls generated by a set of 3145 benign applications (samples were taken from Virus Share and carefully filtered to contain only samples with zero detection rate). The system call collection method for the benign application was the same as for malware system call collection described in [12]. This was done to train CNNs and RNNs for malicious/benign activity classification. It is expected that the number of benign applications with related system calls will increase in the future.

The disproportion of system calls versus the number of applications in case of malicious (16.3/3145) and benign programs (112.56/12110) can be explained by the fact of more "aggressive" and "active" malware behavior compared with legal applications.

2.2. Feature Processing. The data generated by the malware and benign samples were stored in an SQLite database. The system calls sequences were stored in the format provided by Dr. Memory DrSTrace tool (see Figure 2). To evaluate the influence of a number of system calls on classification/detection rate, eight files in csv format were generated with 10, 20, 40, 60, 80, 100, 200, 400, 600, 800, and 1000 of the first system calls in every line in a file, respectively (see Figure 3).

Every system call was assigned with the unique number—a sequence of these numbers represents a system calls sequence produced by the specific malware or benign sample. A special tool was developed by the authors to extract the required number of system calls from the SQLite database.

System calls by benign applications were added to two sets:

(1) Set of six classes (five malicious and one benign)—to be used in the classification accuracy test, i.e., assigning the activity to legal or to one of the five classes of malware programs.

(2) Set of two classes (malware and benign)—to be used in the anomaly detection test, i.e., determining if the activity is malicious or not.

For both of these sets, additional subsets were generated, in which sequences of repeated system calls longer than 3 were replaced with a maximum of 2 repeated system calls (e.g., sequence "4655532" was transformed into "465532") according to the recommendation in [19].

Consequently, 66 sets (files) in total for training and testing were generated. The labelling of sets is presented in Table 1. It is necessary to mention that sets with removed repeated sequences had fewer samples. The main reason is that almost all system calls were identical (e.g., one of the malware samples contained only calls to NtCreateFile).

Datasets AllMalware and AllMalware2 were selected to test if there is any difference in the accuracy as compared with simple ML methods performed earlier [17]. Commercially applicable accuracy of 92.4% was achieved with the Support Vector Machines method, and comparable accuracy of 92.1%. was achieved by the decision tree method.

In this research, deep learning methods were applied to check if they can provide higher accuracy using the same datasets.

Five malware families were selected from AWSCTD for our experiment, each family with at least 100 samples of

```
NtQueryValueKey
      arg 0: 0x158 (type=HANDLE, size=0x4)
      arg 1: 16/18 "Category" (type=UNICODE_STRING*, size=0x4)
      arg 2: 0x2 (type=int, size=0x4)
      arg 3: 0x02c5f094 (type=<struct>*, size=0x4)
      arg 4: 0x90 (type=unsigned int, size=0x4)
      arg 5: 0x02c5f070 (type=unsigned int*, size=0x4)
    succeeded =>
      arg 3: <NYI> (type=<struct>*, size=0x4)
      arg 5: 0x02c5f070 => 0x10 (type=unsigned int*, size=0x4)
      retval: 0x0 (type=NTSTATUS, size=0x4)
```

Figure 2: DrsTrace generated system call sample.

```
7, 10, 10, 9, 3, 3, 3, 9, 10, 10, AdWare
20, 20, 20, 10, 10, 9, 9, 9, 18, 11, Trojan
39, 39, 9, 9, 44, 45, 2, 15, 32, 32, WebToolbar
10, 20, 48, 9, 9, 9, 36, 11, 11, 11, Downloader
9, 18, 21, 22, 9, 9, 26, 9, 18, 23, DangerousObject
10, 40, 26, 26, 29, 29, 13, 9, 16, 41, Clean
```

Figure 3: CSV file sample with all possible classes.

Table 1: Training and testing sets labelling.

Set label	Sequence variations	Comments
AllMalware	10, 20, 40, 60, 80, 100, 200, 400, 600, 800, 1000	Only malware samples
AllMalware2	10, 20, 40, 60, 80, 100, 200, 400, 600, 800, 1000	Only malware samples with no more than two identical sequences in repetition
AllMalwarePlusClean	10, 20, 40, 60, 80, 100, 200, 400, 600, 800, 1000	Malware samples plus and benign samples as additional class
AllMalwarePlusClean2	10, 20, 40, 60, 80, 100, 200, 400, 600, 800, 1000	Malware samples and benign samples as an additional class with no more than two identical sequences in repetition
MalwarePlusClean	10, 20, 40, 60, 80, 100, 200, 400, 600, 800, 1000	Only two classes to train: malware and benign
MalwarePlusClean2	10, 20, 40, 60, 80, 100, 200, 400, 600, 800, 1000	Only two classes to train: malware and benign samples with no more than two identical sequences in repetition

unique family representatives. A family descriptor provided by Kaspersky was used. Table 2 shows the number of unique samples in each training set (family names according to Kaspersky).

SQLite database-based data were converted into easily readable CSV files. The sample data of 10 system calls long sequences file is presented in Figure 3.

The first ten numbers represent a unique system call number followed by the label for the malware family name or legal program ("Benign" label) if it is a benign sample.

2.3. Machine Learning Models Used. The experiment was designed and executed on *Keras* [35], and *Tensorflow* [36] library was used as the backend. The following hardware was used in the experiment environment:

(i) CPU: Intel(R) Core (TM) i5-3570 3.80 GHz (4 Cores, 4 Threads)

(ii) GPU: GTX 1070 (1920 Cuda Cores)

(iii) RAM: 16 GB (DDR3)

(iv) OS: Ubuntu 18.04

Table 2: Data samples count by class.

Class label	Samples count
Trojan	1755
AdWare	4333
WebToolbar	618
Downloader	710
DangerousObject*	105
Benign	2350
Total	9871

*DangerousObject is a malicious software that was detected by KL Cloud Technologies but was not classified exactly.

RNN and CNN methods were used in the experiment. The configuration for the ten system calls can be seen in Figure 4. RNN configuration with LSTM and GRU had three layers: *Input*, *CuDNNLSTM* or *CuDNNGRU*, and *Dense*. CNN had *Input*, *Convolution1D* (with sliding window value of 6), *GlobalMaxPooling1D*, and *Dense* layers. The SVM model was also used to compare the results with previous research [17].

Despite relatively straightforward configuration, the models achieved more than 90% classification accuracy with almost all data samples. The classifiers were trained and

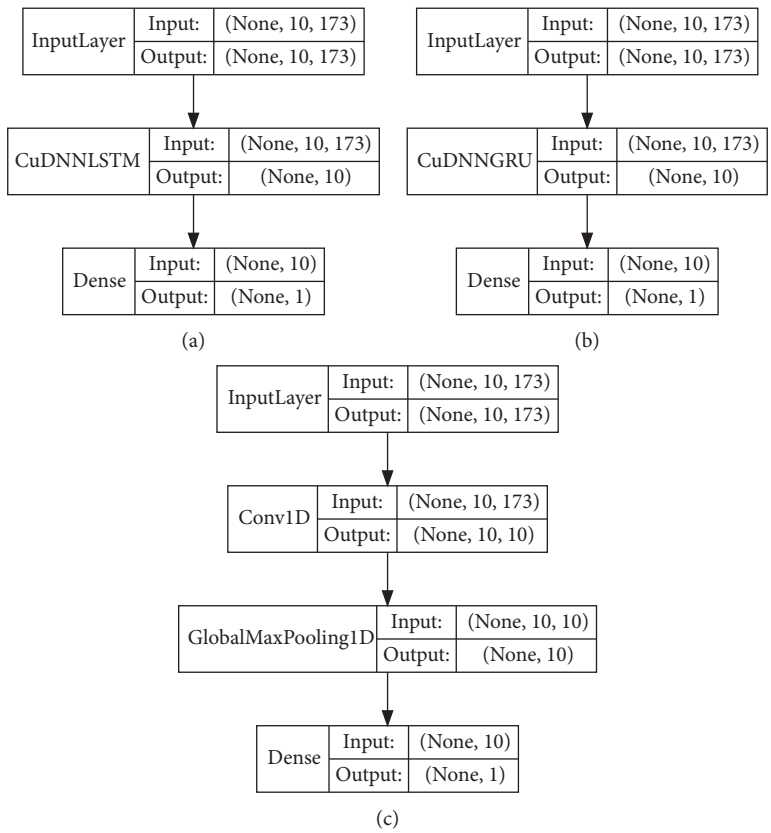


Figure 4: Configuration of training (a) LSTM, (b) GRU, and (c) CNN models for MalwarePlusClean data sample.

tested using a 5-fold cross-validation technique. Cross-validation is a technique for evaluating predictive models by partitioning the original sample into a training set to train the model and a test set to evaluate it. The callback of *EarlyStopping* was used to stop the training process when it did not improve for six epochs. Furthermore, we used *one-hot encoding* to provide data for the training models. Ten system calls samples had unique 173 system calls (see Figure 4 for the value of input shape dimensions). Larger data samples had more unique system calls: for example, 400 had unique 488 system calls. In comparison, the data samples of Kolosnjaji et al. [19] had only 60 unique system calls, which means that our dataset is more diverse.

3. Results and Discussion

This section will cover the results of the following tests: malware classification task, family classification task, and anomaly detection task.

3.1. Results of Malware Classification Task. As stated earlier, the results achieved with DL methods were compared with those achieved through classical ML methods in [17]. The data labelled AllMalware were used in that test. Although the original results [17] have shown that SVM method can achieve 92.4% accuracy with the 100 first malicious system calls sequences, it can be seen that DL methods demonstrate significantly better results with the same dataset (see Table 3). The accuracy is calculated as follows: the method of machine learning is trained with a portion of the dataset (80%), whilst another portion of the dataset is used for testing with the trained model, i.e., data used for testing had not been used for training. Therefore, the percentage of correctly classified records is defined as the accuracy.

CNN achieved 92.8% accuracy on the same length of 100 sequence calls. It has also shown better accuracy for 200, 400, 600, 800, and 1000 system calls sequences than SVM (92.7% vs. 89.6%, 93.0% vs. 87.3%, 93.1% vs. 86.1%, 93.0% vs. 84.7%, and 93.1% vs. 83.2%, respectively). This implies that a practically applicable accuracy (>90%) can be maintained even with larger datasets by applying CNN.

Sequence-based DL methods (LSTM and GRU) demonstrated worse results than SVM—the achieved accuracy was equal to 88.1% and 88.3% on the first ten system calls, respectively. CNN not only demonstrated better accuracy but also achieved a somewhat similar classification time compared with the much simpler SVM model. Similar times were maintained even with a larger data sample. In terms of accuracy, SVM demonstrates degrading results in comparison with CNNs.

3.2. Results of Family Classification Task. Benign samples were introduced next in the training process. As described earlier, two sets were used in tests: with repetitive system calls (AllMalwarePlusClean) and without repetitive calls (AllMalwarePlusClean2). The introduction of a new family

Table 3: Malware-type classification with the help of DL and SVM methods (AllMalware set).

Count	Accuracy (percent)				Classification time (seconds)			
	LSTM	GRU	CNN	SVM	LSTM	GRU	CNN	SVM
10	88.1	88.3	87.3	89.4	0.0000926	0.0000840	0.0000401	0.0000440
20	88.8	88.1	89.0	89.4	0.0001043	0.0000994	0.0000510	0.0000583
40	89.1	90.6	91.2	91.6	0.0001327	0.0001377	0.0000618	0.0000842
60	88.2	90.5	91.2	91.9	0.0001786	0.0001704	0.0000890	0.0000848
80	91.8	91.6	92.3	92.7	0.0002194	0.0002221	0.0001157	0.0000937
100	91.6	91.9	92.8	92.4	0.0002559	0.0002566	0.0001290	0.0001165
200	90.4	91.5	92.7	89.6	0.0004440	0.0004363	0.0003019	0.0002392
400	87.6	90.3	93.0	87.3	0.0009840	0.0008142	0.0006222	0.0006739
600	87.4	91.4	93.1	86.1	0.0023443	0.0023052	0.0016681	0.0015564
800	82.1	88.5	93.0	84.7	0.0043159	0.0037894	0.0023929	0.0022612
1000	75.5	89.6	93.1	83.2	0.0068075	0.0056159	0.0033276	0.0032245

to the training set resulted in a decrease in the accuracy (see Table 4).

The removal of repetitive system calls increased the accuracy of the results. The best accuracy of 93.9% was achieved with CNN and 1000 of the first system calls (AllMalwarePlusClean2 data sample). However, a relatively similar outcome of 93.5% was obtained with only 600 of systems calls, which required much less time for training. As results for 600 and 1000 system calls differ only in the error rate, it can be said that a set of 600 system calls is more preferable for practical applications. On smaller sets, the results by CNN were low (86.9%) but still higher than those by LSTN and GRU (85.8% and 85.6%, respectively).

Figure 5 presents the family classification task results by family in case of a set of 100 system calls.

It can be clearly seen that the number of samples in the training data has a huge impact on the correct classification. WebToolbar, Downloader, and DangerousObject labelled samples have more incorrect label assignments than AdWare, Benign, and Trojan. The lowest classification score has a DangerousObject class—zero. That outcome was expected because Kaspersky itself is not sure about the label, and in our prior research, even the best performing SVM model also generated zero correct classification results for this class [17]. Both models of GRU and CNN classified this family as belonging to the Trojan class. Even CNN model, which generates the best performance (90.0% for that specific data collection), shows that DangerousObject class should be labelled as Trojan.

3.3. Results of the Intrusion Detection Test. Finally, the intrusion detection test was performed, i.e., the applicability of DL methods for determining if an activity is malicious or benign was evaluated (see Table 5). All malicious system calls were merged into one family, and the second family contained only benign system calls. As in previous case, sets both with repetitive system calls (MalwarePlusClean) and with removed repetitive system calls (MalwarePlusClean2) were used.

In this case, the set without repetitive system calls produced comparable results with the full set. This implies that the system calls minimization technique is effective and can be used to achieve better accuracy in family classification and intrusion detection tasks whilst minimizing the model training time.

Accuracies of 94.5%, 94.8%, and 99.3% were obtained by CNN for the 100, 400, and 1000 first system calls, respectively (MalwarePlusClean2). CNN has also shown the best results for all data samples in the two-class classification task (i.e., intrusion detection) of all ML methods used: usable accuracy of 93.2% was obtained even for the 20 first system calls.

In the two-class confusion matrices (see Figure 6), it can be seen that fewer Malware samples are assigned to Benign by CNN as compared with GRU results.

This characteristic is essential in the target field; malignant actions classification as benign must be minimal for the IDS. Benign samples decision is somewhat comparable for the GRU and CNN models.

GRU and CNN models demonstrate outstanding results in the means of the receiver operating characteristic curve (ROC) and area under the curve (AUC) [37]. In Figures 7 and 8, the ROC diagrams with the combination of the AUC values are represented for the MalwarePlusClean samples with the 100 system calls. ROC and AUC are displayed for every fold. Mean ROC and AUC are represented with the blue line.

The best mean AUC value of 0.98 is generated by the CNN model for both classes, i.e., there is a 98% chance that the model will be able to distinguish between Malware class and Benign class. The comparable result of 0.97 is achieved by GRU. High AUC value indicates that both models (GRU and CNN) have good class separation capacity.

3.4. Evaluation of System Call Sequence Size on the Model Training Time and the Number of Epochs Needed to Reach the Saturation. The evaluation of system call sequence size on the model training time was performed on the AllMalwarePlusClean set. Figure 9 presents the training time for LSTN, GRU, and CNN with sequences of 10, 100, 200, and 400 system calls, respectively. It can be seen that the increase in sequence length results in the exponential increase of training time, making extremely long sequences not applicable for everyday use.

GRU training time was equal to 57.7 minutes with the sequence of 400 system calls. The best performing CNN

Table 4: All malware plus clean samples accuracy results. The total of six classes was classified.

Count	AllMalwarePlusClean				AllMalwarePlusClean2			
	LSTM	GRU	CNN	SVM	LSTM	GRU	CNN	SVM
10	77.2	78.3	78.7	79.9	80.1	79.4	80.1	81.2
20	83.0	82.3	83.4	83.6	85.8	85.6	86.9	86.7
40	84.1	84.1	85.1	85.3	87.3	87.6	88.8	88.0
60	85.0	85.8	86.1	86.2	87.5	86.4	88.5	88.2
80	87.1	86.8	88.5	87.7	87.5	88.0	89.2	87.9
100	87.1	86.5	88.2	87.4	87.7	87.9	88.8	87.6
200	85.8	87.5	89.4	86.5	86.2	87.3	89.6	86.3
400	80.9	88.1	89.9	81.9	80.3	87.3	89.6	78.1
600	79.8	89.1	93.2	75.5	77.6	89.4	93.5	74.8
800	68.3	85.3	93.5	73.9	41.5	92.3	93.6	73.3
1000	77.1	89.1	93.6	72.9	64.0	84.1	93.9	72.3

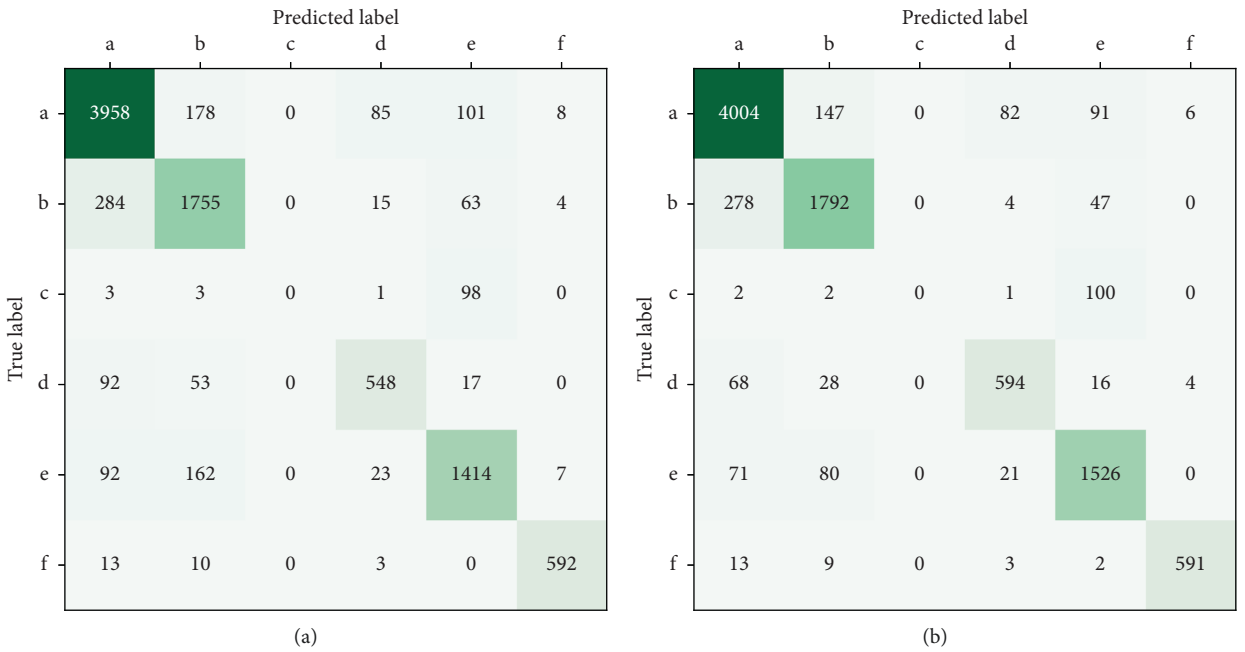


Figure 5: Confusion matrix of the GRU and CNN methods for the 100 system calls sequence of the AllMalwarePlusClean data sample. Class labels: AdWare (a); Benign (b); DangerousObject (c); Downloader (d); Trojan (e); WebToolbar (f).

Table 5: Malicious and benign samples (malware or clean) classification accuracy.

Count	MalwarePlusClean				MalwarePlusClean2			
	LSTM	GRU	CNN	SVM	LSTM	GRU	CNN	SVM
10	87.0	87.4	87.4	87.4	88.6	88.6	88.9	88.4
20	90.0	90.2	91.1	90.2	92.0	92.4	93.0	91.8
40	91.2	90.6	91.8	90.5	92.9	92.4	94.1	92.0
60	91.2	91.1	93.3	91.1	93.2	93.0	93.5	91.9
80	92.7	92.0	94.2	91.5	93.0	93.2	94.5	91.7
100	92.7	92.5	94.3	91.5	93.0	92.1	94.5	91.2
200	90.7	90.8	94.3	88.6	92.4	91.8	94.9	88.9
400	87.0	90.9	94.8	88.4	89.2	92.0	94.8	88.7
600	86.4	88.7	98.5	87.2	85.8	90.1	98.6	87.1
800	84.2	84.7	98.8	86.9	85.1	82.8	98.9	87.0
1000	84.2	70.3	98.9	87.2	83.8	72.4	99.3	87.0

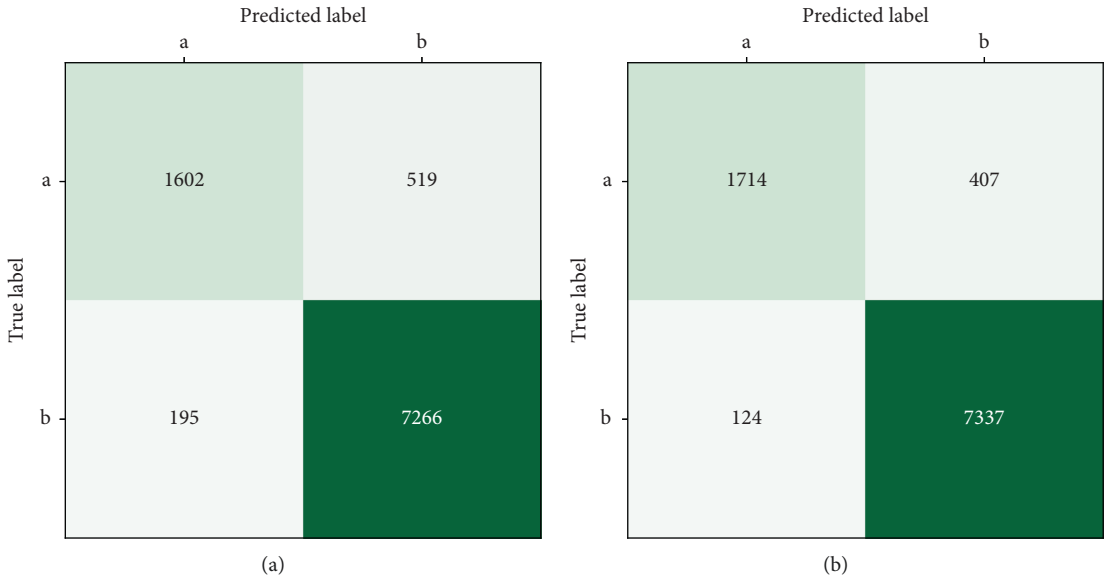


Figure 6: Confusion matrix of the GRU and CNN methods for the 100 system calls sequence of the MalwarePlusClean data sample. Class labels: benign (a); malware (b).

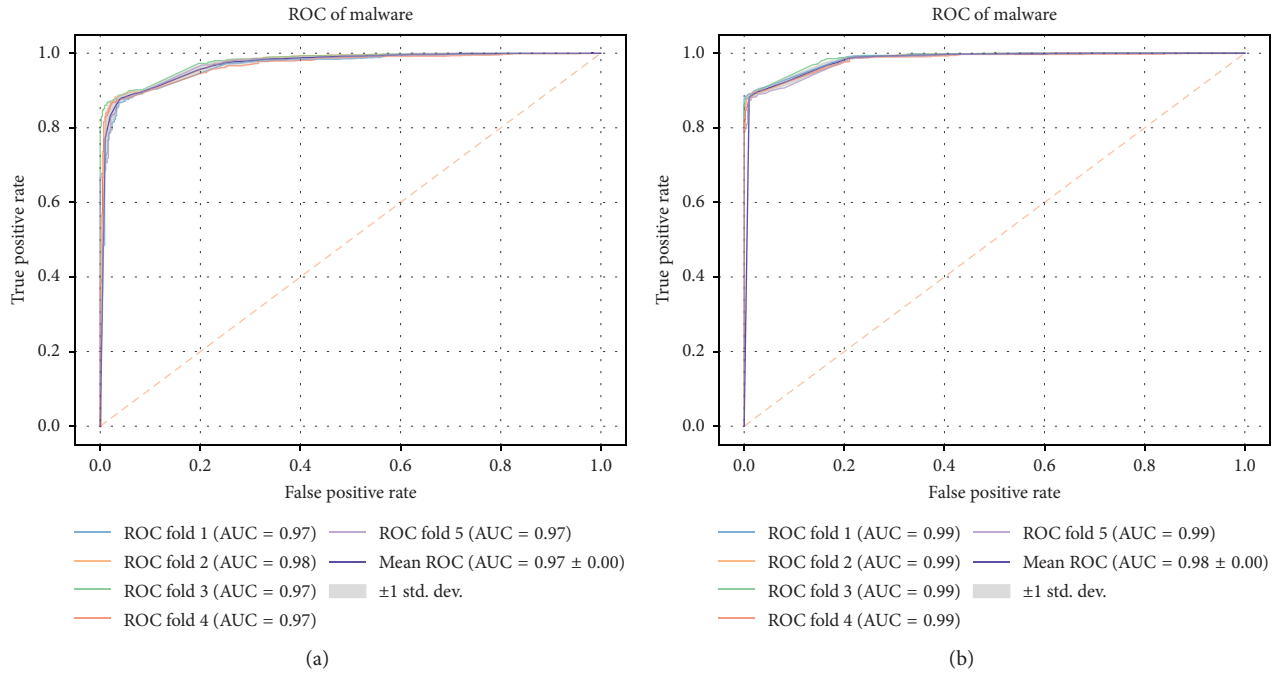


Figure 7: ROC diagrams of the malware class for the 100 system calls of MalwarePlusClean data. (a) GRU. (b) CNN.

model training took 29.6 minutes with the same dataset. In comparison, 100 system calls sequences training time is much faster. For GRU and CNN, it took 4.6 and 3.9 minutes, respectively.

The evaluation of data model size impact on training time leads to the conclusion that using the first 100 system calls sequences is an optimal solution in terms of time and accuracy balance.

The classification accuracy vs. the number of epochs needed to reach the saturation measurement is presented in Figure 10.

As it can be seen, there is a reverse dependency of the number of epochs before saturation on the system call sequence length, e.g., for the top-performing CNN model, 75 epochs are required to train 10 system calls, and only 30 epochs are required for 400 system calls.

The computed equal error rate (EER) [18] values for the DL methods (LSTM, GRU, and CNN) can be seen in Table 6. When comparing models, lower EER means better performance of the model. In our case, CNN shows best values of 9.7% and 4.8% for the Benign and Malware classes, respectively.

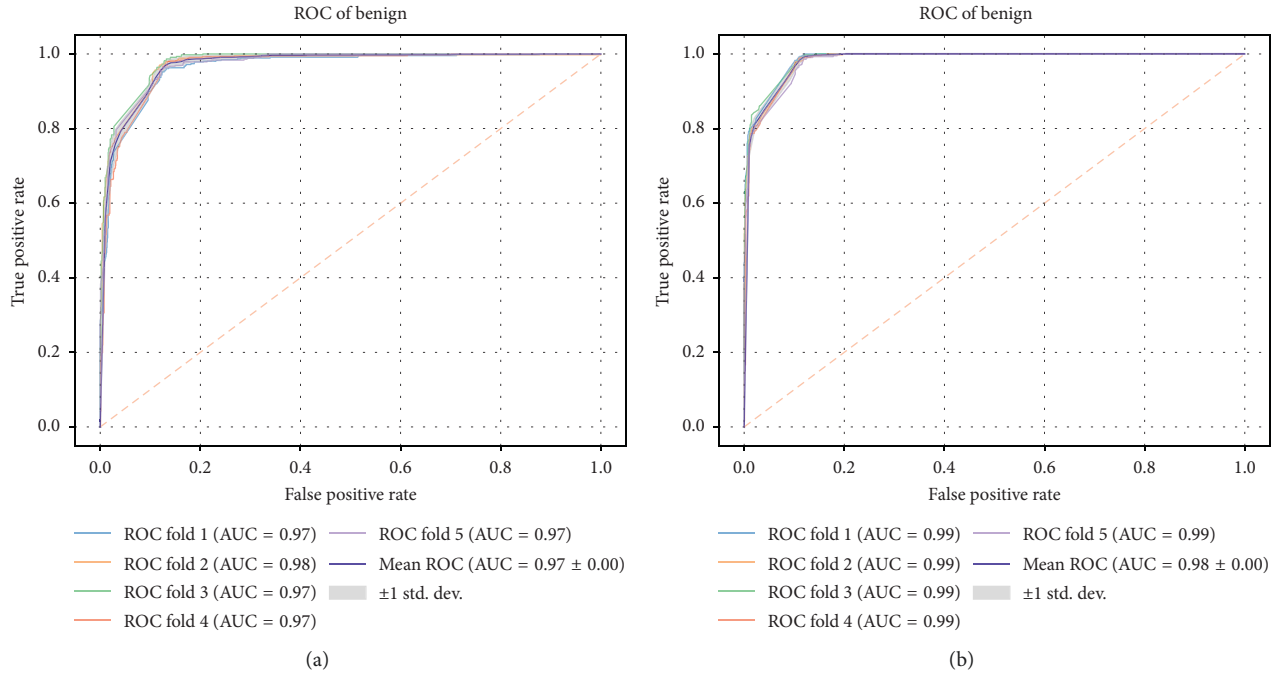
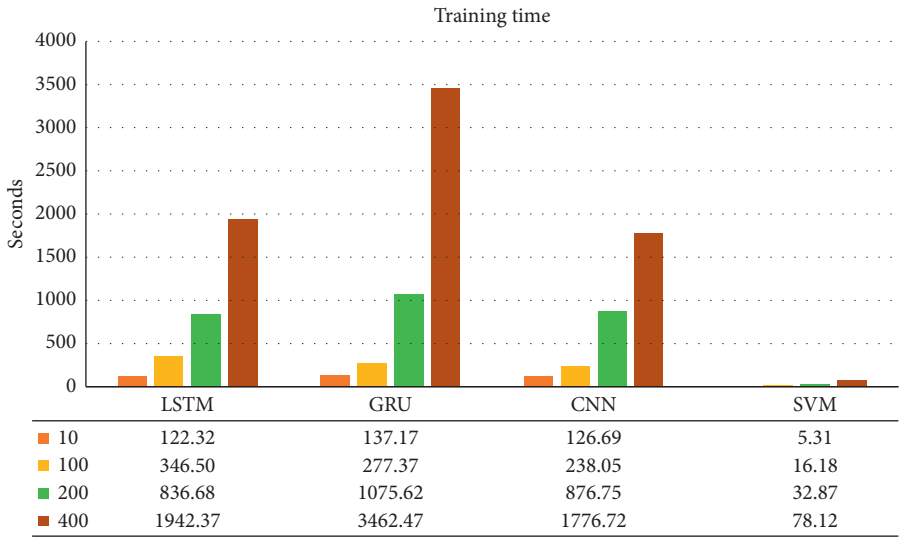


Figure 8: ROC diagrams of the benign class for the 100 system calls of MalwarePlusClean data. (a) GRU. (b) CNN.



	LSTM	GRU	CNN	SVM
10	122.32	137.17	126.69	5.31
100	346.50	277.37	238.05	16.18
200	836.68	1075.62	876.75	32.87
400	1942.37	3462.47	1776.72	78.12

Figure 9: Training time comparison of the AllMalwarePlusClean data collection. 10, 100, 200, and 400 sequence calls as data points.

4. Conclusions

A comparative analysis of DL methods, including LSTN, GRU, and CNN was performed in order to evaluate their efficiency for attack classification as well as their ability to distinguish malicious and benign activity. The analysis was performed on an exhaustive AWSCTD, which includes 112.56 million system calls from 12110 executable malware samples and 3145 benign software samples with 16.3 million system calls. The application of such a set increases the classification and intrusion identification accuracy even with vanilla models by 13–38%, compared with the results achieved by other researchers. Furthermore, model tuning should decrease the FAR even more. In general, the achieved accuracy of over 90% allows the application of DL techniques in hybrid or enterprise-oriented security solutions that combine automatic detection of major part of anomalies, leaving unclear cases for human-expert analysis.

All three LSTM, GRU, and CNN models have reached higher than 90% accuracy while solving a malware classification task with a sequence of 80 system calls. All three models generated improved results over simple NN and SVM models on larger data samples, while the latter demonstrates considerably better training times. The best results were obtained with CNNs with up to 90.0% accuracy while performing family classification task and 99.3% rate while solving intrusion

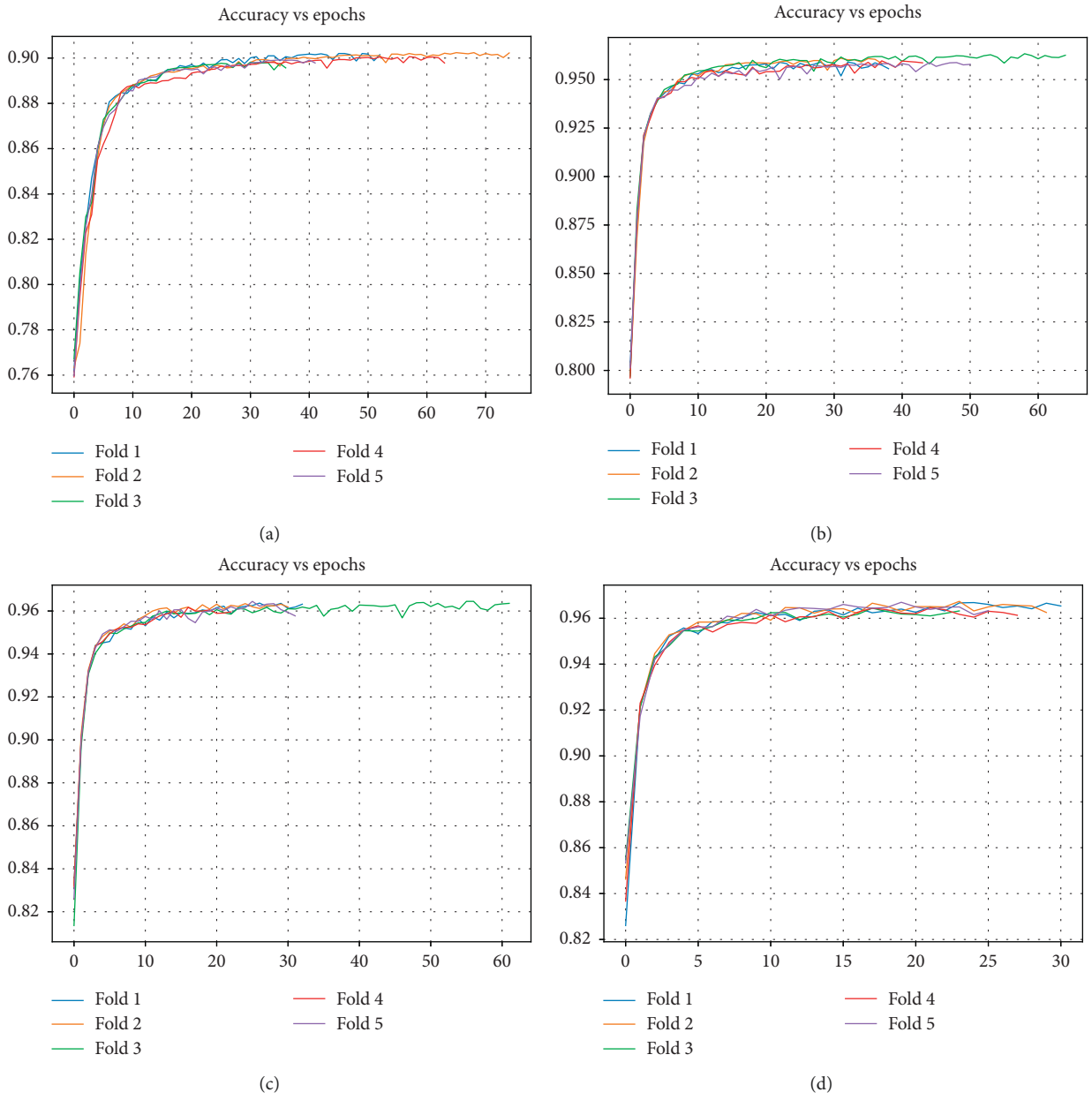


Figure 10: Accuracy vs. epochs needed to reach the saturation while training for the intrusion detection task. Compared CNN models for the 10, 100, 200, and 400 system calls sequences. (a) CNN 10. (b) CNN 100. (c) CNN 200. (d) CNN 400.

Table 6: EER values in percent of the DL methods generated for the MalwarePlusClean of 100 system calls dataset.

DL method	Benign	Malware
LSTM	10.9	7.5
GRU	11.0	8.0
CNN	9.7	4.8

detection task. CNN outperforms sequence-based LSTM and GRU models in all the cases. CNN also shows the best results as compared with EER values of DL methods used. A system calls minimization technique, when repetitive system calls were removed, had a positive influence on all results.

The increase of sequence length resulted in an exponential increase of the model training time, making extremely long sequences not applicable for everyday use. One of the best performing CNN models training took 29.6 minutes, which can be explained by a limited amount of resources on the machine used for the experiments. A reverse dependency of the number of epochs before saturation on the system call sequence length was determined, e.g., for the top-performing CNN model, 75 epochs are required to train 10 system calls and only 30 epochs for the 400 system calls.

Data Availability

The data used to support the findings of this study will be available from the corresponding author upon request after six months from paper publication.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[2] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: techniques, systems and challenges," *Computers & Security*, vol. 28, no. 1-2, pp. 18–28, 2009.

[3] J. Hu, "Host-based anomaly intrusion detection," in *Handbook of Information and Communication Security*, pp. 235–255, Springer, Berlin, Germany, 2010.

[4] R. Bace and P. Mell, *NIST Special Publication on Intrusion Detection Systems*, NIST, Gaithersburg, MD, USA, 2001.

[5] A. Hay, D. Cid, R. Bary, and S. Northcutt, *OSSEC Host-Based Intrusion Detection Guide*, Syngress, Burlington MA, USA, 2008.

[6] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[7] R. P. Lippmann, D. J. Fried, I. Graf et al., "Evaluating intrusion detection systems without attacking your friends: the 1998 DARPA intrusion detection evaluation," in *Proceedings of the DARPA Information Survivability Conference and Exposition. DISCEX'00*, vol. 2, pp. 12–26, Hilton Head, SC, USA, January 2000.

[8] T. Brugger, "KDD cup'99 dataset (network intrusion) considered harmful," *KDnuggets Newsletter*, vol. 7, no. 18, p. 15, 2007.

[9] C. Azad and V. K. Jha, "Data mining in intrusion detection: a comparative study of methods, types and data sets," *International Journal of Information Technology and Computer Science*, vol. 5, no. 8, pp. 75–90, 2013.

[10] K. Berlin, D. Slater, and J. Saxe, "Malicious behavior detection using windows audit logs," in *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security—AISec '15*, pp. 35–44, Denver, CO, USA, October 2015.

[11] StatCounter Global Stats, "Desktop operating system market share worldwide," January 2019, http://gs.statcounter.com/os-market-share/desktop/worldwide/2018.

[12] D. Čeponis and N. Goranin, "Towards a robust method of dataset generation of malicious activity for anomaly-based HIDS training and presentation of AWSCTD dataset," *Baltic Journal of Modern Computing*, vol. 6, no. 3, 2018.

[13] W. Haider, G. Creech, Y. Xie, and J. Hu, "Windows based data sets for evaluation of robustness of host based intrusion detection systems (IDS) to zero-day and stealth attacks," *Future Internet*, vol. 8, no. 3, p. 29, 2016.

[14] M. Xie and J. Hu, "Evaluating host-based anomaly detection systems: a preliminary analysis of ADFA-LD," in *Proceedings of the 2013 6th International Congress on Image and Signal Processing (CISP)*, vol. 3, pp. 1711–1716, Hangzhou, China, December 2013.

[15] M. A. Aydin, A. H. Zaim, and K. G. Ceylan, "A hybrid intrusion detection system design for computer network security," *Computers and Electrical Engineering*, vol. 35, no. 3, pp. 517–526, 2009.

[16] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Computer Science*, vol. 60, pp. 708–713, 2015.

[17] N. Goranin and D. Čeponis, "Investigation of AWSCTD dataset applicability for malware type classification," *International Scientific Journals Security & Future*, vol. 2, no. 4, pp. 83–86, 2018.

[18] N. N. Tran, R. Sarker, and J. Hu, "An approach for host-based intrusion detection system design using convolutional neural network," in *Mobile Networks and Management*, pp. 116–126, Springer, Cham, Switzerland, 2018.

[19] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in *AI 2016: Advances in Artificial Intelligence, LNAI*, vol. 9992, pp. 137–149, Springer, Cham, Switzerland, 2016.

[20] L. Deng and D. Yu, "Deep learning: methods and applications," *Foundations and Trends® in Signal Processing*, vol. 7, no. 3-4, pp. 197–387, 2013.

[21] J. Kim, N. Shin, S. Y. Jo, and S. H. Kim, "Method of intrusion detection using deep neural network," in *Proceedings of the 2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 313–316, Jeju, Korea, February 2017.

[22] S. Pervez, I. Ahmad, A. Akram, and S. U. Swati, "A comparative analysis of artificial neural network technologies in intrusion detection systems," in *Proceedings of the 6th WSEAS International Conference on Multimedia, Internet & Video Technologies*, pp. 84–89, Lisbon, Portugal, September 2015.

[23] H. Debar and B. Dorizzi, "An application of a recurrent network to an intrusion detection system," in *Proceedings of the IJCNN International Joint Conference on Neural Networks*, vol. 2, pp. 478–483, Baltimore, MD, USA, June 1992.

[24] I. Ahmad, A. B. Abdullah, A. S. Alghamdi, N. A. Baykara, and N. E. Mastorakis, "Artificial neural network approaches to intrusion detection: a review," in *Proceedings of the 8th Wseas International Conference on Telecommunications and Informatics (TELE-INFO'09)*, pp. 200–205, World Scientific and Engineering Academy and Society (WSEAS), Istanbul, Turkey, 2009.

[25] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," pp. 1–28, 2016, https://arxiv.org/abs/1603.07285.

[26] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., pp. 3104–3112, Curran Associates, Inc., Red Hook, NY, USA, 2014.

[29] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: encoder-decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, Doha, Qatar, October 2014.

[30] J. J. Kim, J. J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *Proceedings of the 2016 International Conference on Platform Technology and Service (PlatCon)*, pp. 1–5, Jeju, Korea, February 2016.

[31] M. Z. Alom, V. Bontupalli, and T. M. Taha, "Intrusion detection using deep belief networks," in *Proceedings of the 2015 National Aerospace and Electronics Conference (NAECON)*, pp. 339–344, Dayton, OH, USA, March 2016.

[32] Z. Wang, "Deep learning-based intrusion detection with adversaries," *IEEE Access*, vol. 6, pp. 38367–38384, 2018.

[33] VirusShare.com, December 2017, https://virusshare.com/.

[34] VirusTotal, V. T., December 2017, https://virustotal.com/.

[35] F. Chollet, "Keras," 2015, http://github.com/fchollet/keras.

[36] M. Abadi, P. Barham, J. Chen et al., "Tensorflow: a system for large-scale machine learning," in *Proceedings of the OSDI*, vol. 16, pp. 265–283, Savannah, GA, USA, November 2016.

[37] T. Fawcett, "An introduction to ROC analysis Tom," *IRBM*, vol. 35, no. 6, pp. 299–309, 2005.

Hindawi
Security and Communication Networks
Volume 2019, Article ID 8467081, 22 pages
https://doi.org/10.1155/2019/8467081

Research Article

Efficient and Transparent Method for Large-Scale TLS Traffic Analysis of Browsers and Analogous Programs

Jiaye Pan, Yi Zhuang, and Binglin Sun

College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 200016, China

Correspondence should be addressed to Yi Zhuang; zy16@nuaa.edu.cn

Received 3 April 2019; Revised 16 August 2019; Accepted 20 September 2019; Published 27 October 2019

Guest Editor: Surya Nepal

Copyright © 2019 Jiaye Pan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Many famous attacks take web browsers as transmission channels to make the target computer infected by malwares, such as watering hole and domain name hijacking. In order to protect the data transmission, the SSL/TLS protocol has been widely used to defeat various hijacking attacks. However, the existence of such encryption protection makes the security software and devices confront with the difficulty of analyzing the encrypted malicious traffic at endpoints. In order to better solve this kind of situation, this paper proposes a new efficient and transparent method for large-scale automated TLS traffic analysis, named as hyper TLS traffic analysis (HTTA). It extracts multiple types of valuable data from the target system in the hyper mode and then correlates them to decrypt the network packets in real time, so that overall data correlation analysis can be performed on the target. Additionally, we propose an aided reverse engineering method to support the analysis, which can rapidly identify the target data in different versions of the program. The proposed method can be applied to the endpoints and cloud platforms; there are no trust risk of certificates and no influence on the target programs. Finally, the real experimental results show that the method is feasible and effective for the analysis, which leads to the lower runtime overhead compared with other methods. It covers all the popular browser programs with good adaptability and can be applied to the large-scale analysis.

1. Introduction

Currently, the incidents of malware attack occur so frequently as to cause the serious loss of data and property to internet users. Malicious code has presented new forms such as ransomware, phishing, and coin mining. Web browsers are the important sources of malware to infect the target computers. For example, users download and install software bundled with malicious codes from the third-party website, or users encounter phishing attacks and access the fake page, or the web browser loads a web page with vulnerability exploitation codes and triggers the infection of malware [1]. Meanwhile, the incorrect configuration of legitimate applications may also cause the exfiltration of privacy data. For terminal users, web browser is an important application in their work and life. Consequently, it is crucial to inspect the content of web pages in order to create a secure internet environment for computer users.

Moreover with the development and popularity of the Secure Socket Layer (SSL) protocol [2] and its subsequent version Transport Layer Secure (TLS) [3], lots of websites and client applications adopt the HTTPS protocol instead of HTTP [4]. It is based on TLS which can prevent the transmission of data from advertising hijacking and packet manipulation. This also becomes the barrier of security information and event management (SIEM) systems. Although the TLS protocol is very secure in theory, there are many kinds of problems in its practical uses. The software which implements the protocol may have vulnerabilities, such as those of *OpenSSL* [5, 6], and some insecure algorithms and cipher suites may cause the cipher text to be cracked [7, 8, 9]. Moreover, there may be Man-In-The-Middle (MITM) attacks in the authentication process, for reasons such as counterfeit certificates [10], certificate verification bugs or lack of security consciousness [11, 12]. Actually the problems led by improper deployment of TLS protocol are more complicated than we thought, especially in the browser application and HTTP protocol [13, 14]. The security of TLS protocol is continuously improved in the confrontation between attack and defense, which has been

shown in the draft of TLS version 1.3 [15]. Meanwhile, there are many enhanced mechanisms for TLS applications [16], some of which are widely used in practice, such as HTTP Strict Transport Security (HSTS) [17], Public-Key Pinning Extension (PKPE) [18], Certificate Transparency (CT) [19], and so on.

In this situation, the firewall based on packet filtering cannot make deep analysis on the packet content. One better solution is to choose, model, and classify the plain information in the TLS connections with training, such as the handshake fingerprints [20] and then to pick out the malicious traffic in the real environment [21]. This method can only discover the abnormal TLS connections, but fails to identify the web page which contains malicious payload because the packet is not decrypted. Another solution is to interpose a TLS proxy in the communication, which is more general in security software and devices. First, adding a trusted Certificate Authority (CA) in the system and configuring the network proxy, when the browser makes a TLS handshake, it will generate a specific certificate for the request domains, and it is signed with the installed certificates beforehand [22]. In this case, trusted and auditable connections will be established. Actually, this method is very suitable for web browsers but not general since many software implement the TLS protocol in a custom manner, also with the certificate or public key pinning. Besides, proxy will delay the network transmission which can be detected by both the client and server, and the incorrect proxy configuration may cause serious security problems [22, 23]. Function hook is also an approach to network data analysis, but it would interrupt the workflow of the program. Additionally, hooks will meet the challenge of program version diversity. There are also some good tools for manual analysis which depend on proxies, such as *Fiddler* and *mitmproxy* [24, 25]. *Wireshark* is also helpful when the master key of TLS session can be obtained and imported [26, 27], but it depends on the special configuration of the target program and also lacks the full automation. The plugins of browsers can also help to analyze the web page content, which do not have the capability of raw packet manipulation and have compatibility issues. From the perspective of attacks, the aggressive methods have limitations, and they will be ineffective due to vulnerability patches. Nowadays, in that the security of TLS is improved, the attack approach cannot make a persistent traffic analysis. It is common that the browser triggers the vulnerability and malware installation when users are working on the internet; hence, traffic analysis needs to be deepened in order to block the malicious code ahead.

In general, the existing methods are inefficient for the large-scale analysis in real time, which also depend on the modification of the target program or system. The analysis is limited to the decryption and isolated from the data analysis in the system perspective. In accordance with the above situations, we deeply research the mechanism of different browsers and analogous programs. Then, we propose the hyper TLS traffic analysis method, named as HTTA, which attempts to analyze the TLS traffic from a new perspective. It aims at the efficiency, transparency, large-scale, automation, and real-time analysis. The key idea is that it extracts the session information from the process space of browsers and then correlates it with multiple types of data and works in real time, so that further correlation analysis can be performed on the decrypted packets. The proposed information extraction method and data analysis pattern can cover the popular web browsers on multiple platforms and overcome the challenge of real-time analysis, without depending on the trusted certificates and hooking methods.

In summary, the main contributions of this paper are as follows:

Firstly, we propose a new efficient automated method and its corresponding framework for large-scale TLS traffic analysis of the browsers; it is also suitable for programs which have the homogeneous crypto infrastructure. The method collects and correlates multiple types of online data with noninvasive mode, which can perform real-time transparent analysis. It carries both efficiency and security and can be deployed flexibly.

Secondly, we propose a new session information extraction method based on the session cache and the characteristic of low fragmentation heap, which can cover the popular web browsers. Additionally, for coping with the version diversity of programs, the instruction similarity matching method with the constraint of graph path is further proposed to locate the key variable and function, to help extract the relevant session information.

Thirdly, we give a prototype implementation on Windows platform and analyze its essential links. The framework has a simple structure for fast installation and deployment, and it can be used in the further development of the automated analysis system for TLS traffic in the real environment.

Finally, this proposed method is verified by experiments with multiple types of browsers in reality. The experimental results demonstrate that the method can perform precise and efficient analysis with the lower performance overhead and also make no interruption to the workflow of the target program at the same time.

The rest of the paper is organized as follows. The related work is reviewed in Section 2. Section 3 describes the proposed method and analysis approaches in detail. Section 4 shows the key points of implementation. The conducted experiments and results are demonstrated in Section 5. Section 6 gives the discussions, and Section 7 concludes the work.

2. Related Work

There are a lot of research work about the attack and defense techniques of browsers and traffic analysis, and some related work with this paper are as follows.

2.1. Protocol Crack and Attack. Duong et al. proposed the BEAST attack which could obtain the plaintext passed

between the browser and server. It utilized the weakness of cipher block chaining (CBC) mode in the TLS protocol [8]. Rizzo et al. proposed the CRIME attack [7], which tried to guess the key request data such as cookies in the TLS channel exploiting the weakness of TLS compression method. Fardan et al. proposed the Lucky Thirteen attack which used a timing attack against the CBC encryption mode [9]. Padding Oracle [28], BREACH [29], so on are similar attacks. These attacks are complex, most of which need to inject the attack code into the victim's browser and send large requests to the web server. They also depend on the specific protocol versions.

2.2. Proxy and Traffic Analysis. Marlinspike et al. designed and implemented SSLStrip which could intercept the HTTP request before it redirected to HTTPS [30]. Liang et al. discovered the HTTPS deployment problems in CDNs, which could cause MITM attacks [31]. Jia et al. proposed the browser cache poisoning (BCP) attack [32], which amplified the threat of MITM attacks by poisoning the browser cache for persistence after the user ignores the warning of illegal certificates. Sherry et al. proposed BlindBox [33], which was a deep packet inspection (DPI) system based on encryption traffic, whereas a new protocol and encryption scheme needed to be designed. Similarly, searchable encryption is widely discussed in the cloud storage, outsourcing, and so on [34]. But it may have difficulties to fully secure network communications, for example, the encryption efficiency is low, and anyone on the network route can search the content by keyword guessing. Even though, from the perspective of traffic analysis, it is difficult to understand the semantics of the whole communication through some searches only.

Durumeric et al. probed into the impact on HTTPS by security software and network devices [23]. It pointed out that web servers could detect these behaviors, and the interception might weaken the security of original TLS connections. Carnavalet and Mannan designed a framework for analyzing the TLS proxies on the client, which could uncover the security risks introduced by these interception tools [22]. However, these research studies showed that the network proxy might cause the security problems, and similarly malicious code could also disrupt the proxy configuration with the same method. In this paper, HTTA does not act as a TLS proxy so as to avoid raising security problems. It is transparent to the both sides of the communication.

2.3. Memory Data Extraction. Dolan-Gavitt et al. proposed the virtual machine introspection (VMI) framework Tappan Zee (North) Bridge which could analyze the memory data [35]. The keyfind plugin is used to search master keys in the memory, and it tries to decrypt and validate the packet by using the each 48-byte data as a master key. Similarly, Taubmann et al. proposed TLSkex which could also extract master keys of TLS connections based on VMI technology [36], and it used a brute force together with some heuristic approaches based on searching in a memory snapshot. Feng et al. proposed ORIGEN [37], which was applied to get the data structure profiles in the new versions of the software based on the knowledge on its old version. It is helpful to solve one of the problems in our framework. We also propose another solution to solve the location problem of pivotal functions and variables.

2.4. Page Content Analysis. Vadrevu et al. proposed a new web browser with the forensic engine named ChromePic [38], which could record and reconstruct the process of common web attacks based on Chromium. Jayasinghe et al. proposed a novel dynamic approach to detect drive-by download attacks [1], and it can monitor the bytecode generated by a browser in real time with low performance overhead. Studies based on the page content analysis fall on the next step of our research, and some can be integrated into the proposed framework in this paper.

In general, compared with the existing research work, we mainly focus on the efficient automated TLS traffic analysis in the large scale, which will recover the original text in the encryption channel and make further data analysis. Moreover, we unwrap the network packets by correlating multiple types of runtime data, and will deal with the effect of decryption and the real-time analysis problems. The proposed method in this paper is applicable and scalable, which can be easily deployed in practice.

3. Method Description

3.1. Architecture Overview. Traffic encryption mechanisms improve the capability of data protection, while they weaken the security data analysis system. To address this, flexible and efficient schemes are needed for TLS traffic acquisition and correlation. Two questions should be considered which also motivates the research. One is: how to defend the vulnerability exploitations and malicious codes before the browser parses and renders the page? The thorough method is to analyze the content of network traffic continuously and extract the traffic characteristics to the host or network intrusion prevention system. The other is: how to improve the efficiency and reduce the impact on the target system while analyzing? It preferably needs the noninvasive approaches to ensure the security and real time.

The overall architecture of HTTA is shown in Figure 1; it collects multiple types of data associated with the target programs in the hyper mode, for example, from the kernel space, hypervisor, or hardware device. The acquired data include processes, threads, active network connections, file operations, and TLS sessions on the target operating system. The network traffic is also collected and filtered, which can be performed in a bypass mode if there are no packet interception requirements. The noninvasive method is used which means it does not modify the executable module and configuration of target program, and also it does not intercept the original workflow which may lead to the failure of communication. Transparency means that the target program and remote server cannot directly detect the presence of the analysis. There is another notable point, as seen from Figure 1, that the dotted lines denote the appropriate interactions with the target system in the special case. For

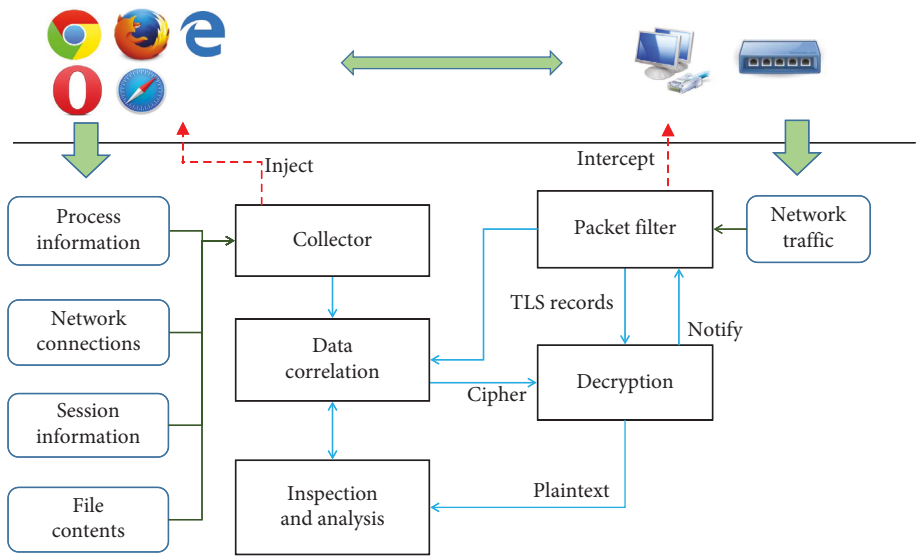


Figure 1: Architecture of HTTA.

example, inject a thread into the space of target process to accomplish aided task. For the most cases, it is not requisite.

The process information contains the process name, the executable file path, loaded modules, and process/thread environment block (PEB/TEB). According to the information, we can determine the target program together with its version and crypto infrastructure. The network connections mainly include the IP addresses and ports of target process referring to the TCP/IP protocol, which can be directly correlated with the network packets. The above data are acquired from the kernel space of the operating system; there have already been lots of research studies about that. The session information includes the essential key and related parameters for decrypting the traffic, which exist in the memory space of the target process, and we call it *TLS session information* (TSI). Actually, the former information is also in the memory space, which is managed by the system kernel. File handles refer to the files opened by the target program, and the file content can be further correlated with the decrypted network packets. The data correlation module first correlates the target program with the TLS traffic at the level of TCP/UDP stream, and later the correlation results will be sent to the inspection module for decryption and analysis. The packet filter module identifies the needed network traffic and can also perform preliminary analysis to extract the fingerprint of the crypto library. Besides, as denoted by the dotted line in Figure 1, the module can decide whether or not to pend packets of the specific stream for a while until the extraction module obtains the associated TSI. This will be discussed below in this paper.

For further discussion, the proposed method also has good extensibility and compatibility. It is suitable for not only web browsers but also other programs, while the browser programs are widely used and relatively stable. We can construct the unified TSI extraction patterns for browsers, but it is difficult to cover other unknown programs, and then individual analysis is needed in advance. HTTA is not limited to the platform; it can be applied to Windows, Linux, and others. It supports all the TLS versions, including old SSL and the newest TLS 1.3. Because the essence of symmetric encryption mechanism has not been changed, the handshake protocol changes enormously. The encryption mode such as stream cipher and block cipher has no influence on the method. Moreover, HTTA is convenient for engineering extension and deployment, for example, modules can be moved from the user space to kernel space, or to the outside of the system. On the other hand, it is helpful to migrate between operating platforms, for example, avoiding the platform interdependency of packet capture and memory access module. The number of external interfaces is reduced in order to keep it independent. In fact, we only need two interfaces which are memory read and network access.

3.1.1. Aided Packet Interception. In the scenario of all the traffic can be decrypted in time, to ensure that we additionally introduce an additional interaction procedure between the packet filter and information extraction modules, as shown in Figure 1. Because in particular situations, for example, the configuration of cache time has been modified, the session information will not be cached for a long time. When the packet filter module detects that the TLS handshake is finished, it notifies the information extraction modules and queues the next packets at the same time, as are shown in steps 1~3 in Figure 2. Information extraction module then extracts the required data, after which it notifies the packet filter module again, as are shown in steps 4~5 in Figure 2. The packet filter module continues to forward the previous packets in the waiting queue, as are shown in steps 6~7 in Figure 2. Therefore, under this mechanism, it is for sure that information extraction modules can obtain the session information through the control of network packet forwarding. With the possible little cost of network delay, we should avoid the TCP timeout.

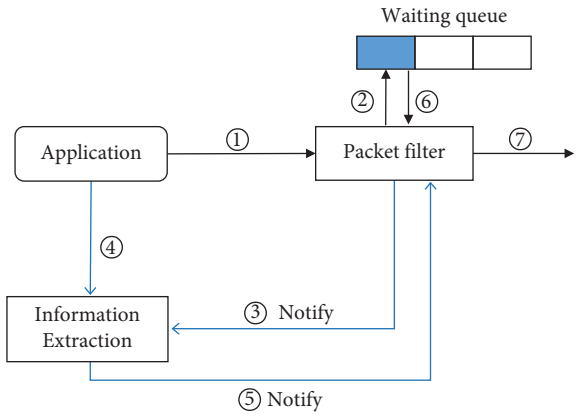


Figure 2: Additional interaction of the extraction and filter module.

In this mode, the session information and subsequent packets will be transferred to the decryption and analysis module for further process. Another waiting point can be set when packet filter module waits for the notification of analysis module and then decides whether to drop the target connection or not. It does not occur in all the packets. Because a TLS record often consists of multiple TCP packets, the decision can be determined in the last packet of the record, even in the last packet of stream, for better optimization. Moreover, if HTTA works in the offline analysis mode, there would be no interception delay.

3.1.2. Deployment and Operating Modes. For different application scenarios, there could be three deployment modes of HTTA. In the first scenario, the analysis framework is installed on the local system, and it can be extended based on further development or integrated with other security analysis components. The second scenario is in the virtualized environment; the analysis framework is deployed in the hypervisor component and then can analyze the TLS traffic of programs on the VMs. In the third scenario, it is deployed on the access point of internal network just as the firewall.

As shown in Figure 3, in the first case, HTTA is divided into the user mode and kernel mode module. The TSI extraction and packet capture module are in the kernel mode; data decryption and analysis can be in the user mode, which is convenient for further development according the specific requirement. In the second scenario, all modules are stayed in the hypervisor, and it is out of the band and fully transparent to the target system. Furthermore, data from different VMs can converge into one process node in this case. For the above cases, the network packets are collected in the kernel of the target system, which would intervene the target communication to some extent, but it only forwards the packet without unwrapping and wrapping. In the third scenario, a proxy module should be deployed on the target computer, and then the TSI obtained by the proxy module will be sent to the local firewall together with the network traffic. At this moment, the packets can be collected by port mirroring of the network device, which does not introduce any direct interceptions. For different deployment modes, the core logic of HTTA is the same, so the prototype below concentrates on the first deployment mode.

Additionally, HTTA can be configured with audit mode or interception mode which is suitable for traffic audit and real time analysis. In the audit mode, the framework only considers the integrity of TSI. While real time is necessary in the intercept mode, it can block and replace the target content. For accomplishing the TLS packet filtering according to the designed framework, there are several influence factors. First, the TSI extraction works based on the dynamic environment, not on the offline memory dump, and the precise should be ensured. Second, TSI and network packet could be correlated since they are obtained in the separated process. Finally, correlation and decryption should be finished in a short time for real-time analysis. We will give further description and discussion in this paper and then analyze and validate that by real experiments.

3.1.3. Challenges and Countermeasures

(a) *TSI Extraction*. There are some searching techniques in the memory, such as Magic number and debug symbols. However, it will cause high performance overhead when searching in the large address range. To overcome this, we propose an optimized method. Some browsers support log mode which can output TLS master key into a file [26], but this should modify the runtime configuration, and it is not general. Finally, we propose an efficient method of extracting TSI in the process memory of browsers, which covers all popular browsers. It will bring convenience to information extraction together with data structure analysis.

(b) *Version Change*. The minor version of browsers may change frequently, and it will cause the binary layout changing because of recompilation. The critical data extraction depends on the reverse engineering of target binary program. In fact, most applications including web browsers are not obfuscated. HTTA only depends on several binary features, if we can locate the address of some functions and then can extract TSI rapidly. We can choose some stable functions that reference the target variable, and expect they will be invariant if the program version updates. We adopt the method of semiautomation to solve the changing problem of the version with consideration of practice.

(c) *Real Time and Overhead*. In HTTA, the TSI extraction thread runs in the kernel space or out of the system, and it will continuously access and control process memory better. The TSI extraction module cooperates with the packet filter module, and it works when the connection is established and stops if the connection is closed; it can reduce the rate of CPU usage. On the other hand, we can properly delay the speed of packet forwarding by the packet filter module, and consequently extend the duration

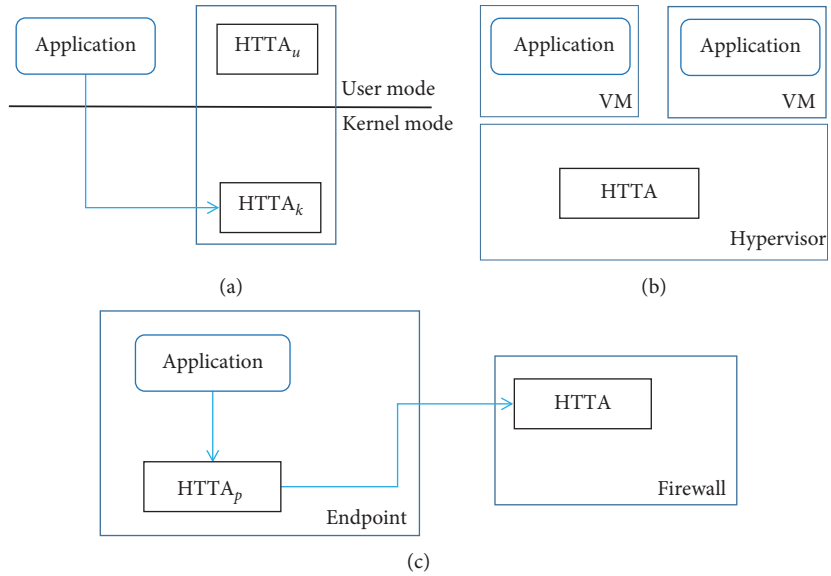


Figure 3: Different deployment modes for HTTA.

of target TLS connection. Then, the extraction modules will have enough time to obtain the session information.

3.1.4. Security Issues and Analysis. No additional trusted root certificates are installed in the system; therefore, the related security risks do not exist, and, for example, the corresponding private key is stolen or cracked. The proposed solution only inspects the network traffic in real time without preserving the data, which will be deleted after the analysis. The potential risk is that the decrypted data can also be accessed by the malicious code if it would be. In this case, we can migrate the analysis code into the kernel which can prevent most of the nonrootkit malware. On the other hand, the memory protection method can also be used such as Intel Software Guard Extensions (SGX). In fact, if the malware comprises the target system, it may directly extract the sensitive data from the target application or other resource storage more easily.

Meanwhile, the solution does not directly intervene the execution flow of target program, and it reduces the impact of uncertainties. Network packet forwarding is the only intervention. When packets are collected by the kernel module, the module should be minimized and safely developed. In the audit mode, it has a minimal effect on the network communication of target program. In the interception mode, additional daemon threads can be created to watch the process of packet forwarding, so that it can handle the connection timeout in time due to the exception of the analysis system.

3.2. Approaches of the TSI Extraction. The procedure of TSI extraction contains two parts: one is to extract the classes or structures which are related to the target TSI, and the other is to extract the internal member information of the structures above. We can extract these data from the process memory space of programs, but need to obtain the structure location and member offsets of TSI, depending on the binary or source code analysis preliminarily. For example, we can obtain the TLS session structure according to the source code of *OpenSSL*, which is defined as *SSL_session*. It contains the master key which is the critical data for the encryption and decryption according to the TLS specification, and then we can decrypt the packets based on that. So, the main goal in the extraction is the data structure like above.

According to the stipulations of the RFC documents, TLS protocol consists of key exchange and data encryption transmission, and the latter uses symmetric encryption mechanism. In other words, the both ends of communication hold the symmetric key information. The client end also holds the key information while the connection is alive. Moreover, the TLS protocol supports the session resumption for improving efficiency, using *session ID* (SID) or *session ticket* to identify the connection. So, the client program would cache the session information in the memory for resumption with the above ways.

3.2.1. Overview of Typical Browsers. We focus on the browser programs on the Windows platform, which is widely used and vulnerable in the developing countries. There are abundant applications but a few web browsers with high market penetration on personal computers, which are *Chrome, Firefox, IE, Edge, Safari,* and *Opera* [39]. Other browsers, such as *Maxthon, 360*, and *Sogou*, are all built on the core engines above.

First, although the network modules of browsers are slightly different, they all adopt the multiprocess architecture in the newest version. For *Chromium*-based browsers, the main process is in charge of network communication, so the TLS data transfer is in the separated process, and *Firefox* is similar. The development of *Safari* on Windows has been stopped. The version on Mac OS uses a separate process to

process network communication. Unlike other browsers, the network data are processed separately in each render process in *IE* series.

Second, the web browsers have different TLS implementations. Chrome and Opera based on *Chromium* use the *BoringSSL* which is a fork version of *OpenSSL* [40, 41], and *Firefox* maintains its own implementation named network security services (NSS). *Safari* uses Apple Secure Transport and *coreTLS* libraries. *IE* and *Edge* have part implementation in the *schannel* and *wininet* library. Most of the other browsers are the integration of the above-mentioned facts. In fact, other forks of *OpenSSL* such as *LibreSSL* are similar in the aspects of this paper [42]. The basic information of TLS implementation of some web browsers on Windows is shown in Table 1.

As the analysis shown above, although there are many differences between the TLS implementation for kinds of browsers, the TLS session information is managed by the specific module loaded in the memory. Therefore, we can extract the session data from the process memory space. It is also suitable for other programs which are built on the analogous crypto libraries of TLS.

3.2.2. Data Extraction. Due to the session resumption mechanism, modern web browsers always link the session structures together and cache them in the relative fixed memory region. Then, these chained structures are referenced by some global variables, in which the popular browsers and derivatives all have such characteristics. So, if we can locate the addresses of target global variables, then the session information can be extracted rapidly by traversing the hierarchy structures. We describe the TLS cache management for the main browsers below.

Chrome does not use the internal cache method provided by *BoringSSL* or early in *OpenSSL*. It manages the TLS cache externally, which defines the class *SSLClientSessionCache* [40]. When the TLS handshake initializes, the browser tries to look up existing session in the cache list and insert new session into the cache list when handshake finishes. The cache structure is *ssl_session_st* which contains established time, resumption information, cipher parameters, and so on. The cache list uses *std::list* to manage the session structures. Class *SSLClientSessionCache* is referenced in the class *SSLContext* which is defined as a singleton. This implies that the *SSLContext* pointer is stored in the binary as the global variable. *Opera* and other browsers which are built on *Chromium* have the same case.

Firefox defines the static pointer cache in *sslnonce* file [43], and it points to a double-link list structure. The structure is *sslSessionID* which contains accessed time, session ID, master key, and so on. It should be noticed that the master key is not stored in the form of plaintext and encrypted through the Public-Key Cryptography Standards (PKCS). So, we need to obtain the symmetric key used by PKCS beforehand and then decrypt the extracted session information.

Safari defines the variable *_gSessionCache* in the Security and coreTLS libraries, which points to a double queue. The cache structure also contains the master key and session resumption information.

The TLS interfaces of *IE* and *Edge* are encapsulated in the *schannel* library, among which the *SslContextList* variable points to the TLS cache information in the process memory of the browser. When the TLS handshake between the client and server is finished, the system derives the read key, write key, and other parameters from the master key and then stores them into the C*SslUserContext* class which is linked by *SslContextList*. Meanwhile, the link list *GlobalObjectList* and *GlobalServerInfoList* in the *wininet* library caches the current socket information and also have relations with *SslContextList*. A sketch is shown in Figure 4, the needed data is stored in the structure *CSecureSocket* and *CSslUserContext*. Therefore, we can get precise ports and key information in the memory for the *IE*, *Edge*, and other *IE* core-based browsers.

As known from the analysis, the popular browsers all have the global variable which points to the TLS session cache information. So, if we can locate the variable first, all existing TSI can be traversed rapidly.

Additionally, if the session cache data is flushed or the lifecycle of session is too short, we may not obtain the corresponding TSI in time. As a supplement to satisfy the special requirement, we also propose the rapid TSI extraction method based on the *low fragmentation heap* (LFH) mechanism [44], to locate the target in the limited memory region. The operating system provides a special memory management scheme for the memory allocation with small sizes which is named as LFH on Windows. Actually, some applications or libraries also have the similar custom implementations, such as Firefox, and we all call it LFH here. The key of LFH is that the similarly-sized memory blocks are allocated from the same memory bucket by using the bucketing scheme. As shown in Figure 5, when the allocation with specific size is committed, the memory block will be allocated via the corresponding bucket from the preallocated memory chunks. For the structure that contains the TSI, the allocation size belongs to the scope of LFH. The size of target structure nearly remains stable in different versions of the same program and is easy to obtain. When the version of target program is known, the allocation of the structure that stores TSI will be bound to the certain bucket. For example, as shown in Figure 5, the size of target structure is 400 bytes and allocated from bucket #25; then, we only focus on this bucket continuously. Therefore, when the TSI cannot be obtained by the method above, we can extract the fields such as port, session ID, or session ticket from the network packet and then match the target structure in the preacquired memory blocks of the corresponding bucket. It should be noted that the LFH bucket should be activated before it takes over the allocation of the corresponding size for the LFH of operating system. The bucket is activated if the number of allocations for the bucket allocation size has reached a small value, and it can be easily satisfied for browser programs. It can be forcedly activated in the worst case by injecting the specialized thread, as shown in Figure 1.

Furthermore, if the target structure that contains TSI is unknown, and the reverse engineering and debugging are difficult to perform on the target program, the brute force

Table 1: Overview of related modules of browsers on Windows.

Browsers	Number of network processes	Related libraries
Chrome	1	chrome.dll
Firefox	1	nss3.dll/softokn3.dll
IE	≥1	schannel.dll/wininet.dll
Edge	≥1	schannel.dll/wininet.dll
Opera	1	opera_browser.dll
Others	1 or ≥1	chrome.dll or schannel.dll/wininet.dll

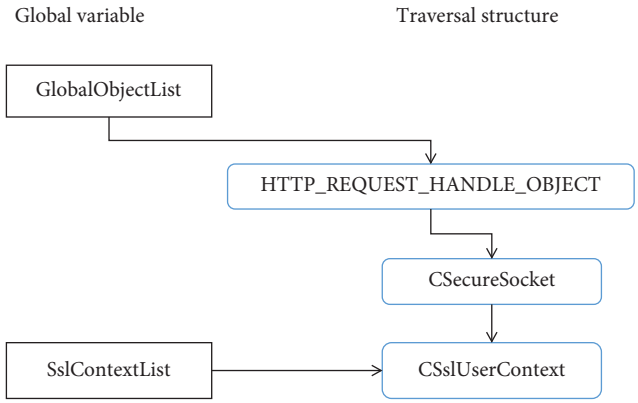


Figure 4: Extraction by the traversal for IE.

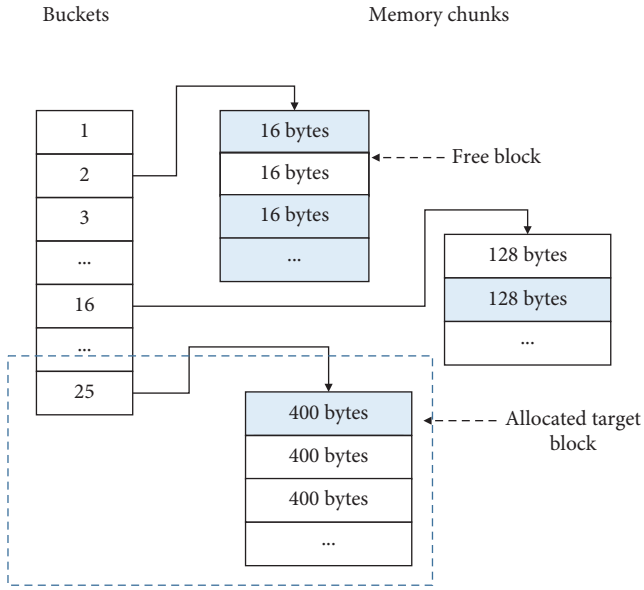


Figure 5: Example of memory allocation via the LFH.

method can be used first to determine the size and location of the target structure [36]. Then, the TSI can be rapidly obtained according to the LFH for the subsequent analysis in the wide range.

3.2.3. Locating the Variable and Structures. We divide the acquisition of global variables that point to the sessions into two cases, one is that the binary program has abundant debug symbols, and the other is contrary. If the debug symbols of the binary program can be accessed, such as *IE*, *Edge*, and *Firefox*, we could quickly obtain the offset value of the target variable according to the debug symbol. In that case, the impact of version change can be reduced. For the latter case, we should extract the offset to the module base by reverse engineering. One simple method is to locate the target address by the reference of constant strings; it is effective in some cases, but not universal.

Then, we introduce the semiautomated method to ease the burden of reverse engineering because the variable can be located through the functions which reference it. When the address of related function is known in one version of the program, we could obtain the similar information from other versions. Two approaches can be applied, one is the graph matching based on the *control flow graph* (CFG) of the target function [45], and the other is proposed here, named as *instruction similarity matching with the constraint of graph path* (ISMCP). The former is widely used in the research of malware detection and binary program similarity detection [46, 47]. But in this paper, we only find the specific function and do not make comparisons on the entire binary program. It can make an accurate match when the CFG of target function has few changes in the new versions. The latter chooses the key path of the CFG to calculate the path similarity between different versions, with the purpose of picking out the target function when the CFG has major changes. It is fuzzy compared with the former. In the first place, both the two methods statically disassemble the target program and generate the CFG for all functions of the program.

(1) Locating the function by CFG matching. CFG is a directed graph, $G = (V, E)$, where V is the vertex set of the graph and E is the edge set of the graph, $E = \{v_i, v_j \mid v_i, v_j \in V, v_j$ may execute after $v_i\}$, the vertex in the graph indicates a program basic block which has no jump instructions. Graph G contains all the possible execution paths of the program. The CFG analysis is widely used in compiler optimization and program analysis. Here, this method builds and models the CFG of target function and then locates new target based on the graph isomorphism check.

Definition 1 (subfunction CFG (SCFG)). The CFG G_f of subfunction F is a tuple.G = (V, E, VE, VX, LABEL_V, LABEL_E), where V is the set of basic blocks, $E \in V \times V$ which is the set of edge, VE denotes the entry block of the function, and VX is the exit block. LABEL_V is the label function of vertexes, which maps each basic block into a

positive integer. Similarly, LABEL_E is the label function of edges.

Then, we define LABEL_V as LABEL_V = SIZEOF $(v)\mp r$, $v \in V$, where the function SIZEOF is used to calculate the total instruction bytes of the basic block and r is an error parameter which is introduced to extend the match range. We do not give the definition of _E, in order to reduce the matching restrictions. Furthermore, one can limit the vertex number of graph to increase the accuracy in the matching, or match with the subgraph G_f' instead of the graph G_f to increase the coverage in the other versions of target program.

When we have the CFG G_{f1} of target function in the specific version of binary program, then attempt to find G_{f2} in the other version, G_{f1} and G_{f2}, is isomorphic. Graph isomorphism is a nondeterministic polynomial time (NP) problem, but CFG has some particular favorable conditions. It has fixed entry vertex, and each vertex has at most two outgoing edges, while the jump tables are an exception. Actually, the case of jump tables can be recognized easily. The VF2 algorithm is adopted [48], and the computation time based on CFG matching will be short in practice. When the change between different versions is slight, the CFG match method can locate the target function precisely. But the graph isomorphism restricts the relationship of edges, and it would expose the limitations when some edges of the CFG of the corresponding function are broken in the new version of program.

(2) Locating the function by ISMCP. For the purpose of locating the target function in the new version when the CFG of target function has major changes, we propose the ISMCP method which can be helpful to extract the target function.

Definition 2 (path in SCFG). Path P is an order $v_1, v_2, \ldots, v_k$, where v_i belongs to vertex set V, v_i, v_{i+1} belongs to set E, $1 \le i \le k$, and vertexes in the order are different from each other. k is the path length.

Definition 3. Path similarity,

$$\mathrm{PATH\_SIM}(S, T) = \sum_{i=0}^{\mathrm{PATH\_LEN}(s)} \frac{\mathrm{LCS\_LEN}(\mathrm{STR}(vs_i), \mathrm{STR}(vt_i))/\mathrm{MAX}(\mathrm{LEN}(\mathrm{STR}(vs_i)), \mathrm{LEN}(\mathrm{STR}(vt_i)))}{\mathrm{PATH\_LEN}(S)}, \quad (1)$$

where S and T are two paths, PATH_LEN is the function of calculating path length, and $\mathrm{PATH\_LEN}(S) \le \mathrm{PATH\_LEN}(T)$, $vs_i \in S$, $vt_i \in T$, $1 \le i \le \mathrm{PATH\_LEN}(S)$. LCS_LEN is a function which calculates the length of the longest common string (LCS) [49], LEN calculates the length of the string, MAX returns the maximum value of arguments, and function STR converts the basic block into a byte sequence.

While the program version updates, some codes of the specific function may also change in the new version, which would cause the change of CFG. As shown in Figure 6, the two CFGs are not the same, but there is the same execution path, *020112*. If the path length is short, we can also measure the similarity of two paths by computing LCS, instead of direct comparison. We expect that the function of the new version preserves part execution features compared with the old one, and they can be found by the matching of instruction sequences. If the similarity between the new and original functions is low, we consider that they do not have the correspondence. In that situation, we need to choose the function in the new version as the template instead, expect to still find the corresponding one in the subsequent versions of the program.

The detailed steps of ISMCP are described as follows:

Step 1: selecting the matching paths. First, we choose some matching paths in the target function of the initial version, which can represent the vital execution flow of the function; meanwhile, take the entry block of the function as the starting point of paths and confirm the path length k (suggest $k \le 12$). If k is too large, the capability of matching will decrease, and the path number will grow exponentially. When paths are confirmed, we continue to label the edges of each path, and the label function is as follows:

$$\mathrm{LABEL\_}E(v_i, v_{i+1}) = \begin{cases} 0, & \text{if the jump condition is FALSE,} \\ 1, & \text{if the jump condition is TRUE,} \\ 2, & \text{others.} \end{cases} \quad (2)$$

If the jump condition is FALSE from block v_i to v_{i+1}, label v_i, v_{i+1} is marked as 0 and marked as 1; on the contrary, in other cases, it is marked as 2. Finally, we get the set of path templates $\mathrm{PT} = \{P_i \mid 1 \le i \le k\}$. Figure 6 shows two same paths which belong to different CFGs.

Step 2: generating the byte sequences. We should convert the basic blocks into integer values. Before this, we map all the instructions of the architecture to 16-bit integers, and build a mapping table in advance. Then, instructions of each basic block will be translated to the byte sequence, as shown in Figure 7. This can preserve the semantic feature of the target function and ignore the influence of specific registers and memory addresses.

Step 3: calculating the path similarity. Export all CFGs of functions in the target executable and traverse all the CFGs successively based on depth first search algorithm. In the traversal process of each CFG, we check the edge label of each path until the end of the path. For one path, if all the label values are matched with the

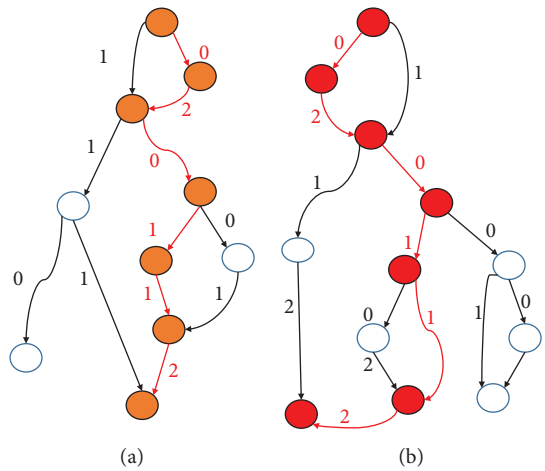


Figure 6: Two paths of different CFGs having the same labels.

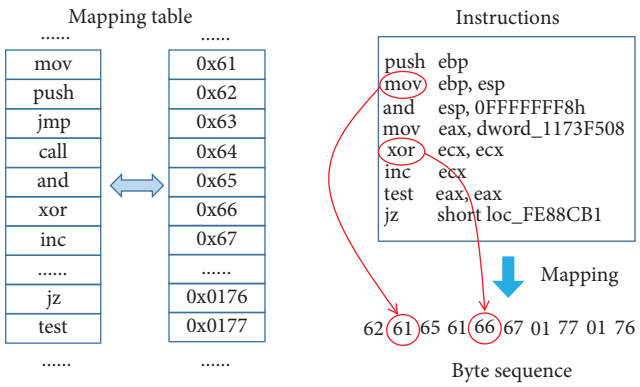


Figure 7: Example of instruction mapping.

path template, we calculate the path similarity of both. As mentioned above, the LCS method can also be used in the matching. Otherwise, we continue to traverse until all path templates are checked or the entire CFG is walked through. The detailed procedure is shown in Algorithm 1.

Step 4: sorting all similarity values. When all the CFGs in the target binary program are traversed, lots of similarity values will be generated. Then, we sort these values in descending order and show the results together with the address of each function. Finally, we pick out the top similarity values of each path template for further analysis.

We expect that the above method can assist with the rapid locating of the target function when part of the code changes in different versions of the program, especially when the structure of CFG changes greatly. The complexity of path similarity generation algorithm is $O(M*N*2^k)$, where M is the total number of the CFG in target program, N is the vertex number of the CFG, and k is the length of path. In reality, the CFG structure is sparse, so when k is small and the root node is fixed, the method can be executed efficiently.

3.2.4. Offset of the Structure Member. Another problem is to extract the members of structure when the starting address of chained structures is located. It is easier for programs which are open source or have abundant debug symbols. One can clearly see the internal variables and understand the logic, or calculate the member offset in a structure according to the variable type of the source code. For the common binary program, static manual reverse analysis is a time-consuming job. Similar to the location of the target structure, we can first determine the function that references the member or parses the entire structure. Then, the specific version of target program is selected as the initial template to obtain the information of other versions. On this basis, we can rapidly examine the changes of member offsets when the program version updates.

Another aided method is to debug and analyze the target program dynamically. So, we can develop and set a local TLS server program based on the modified *OpenSSL*, so that the TSI of each connection would be known, and it can be replayed in addition. The target program is then started and connected to the local server. When the TLS handshake completes, the address of corresponding TSI is also determinate. Then, in the memory space of the browser, we can recursively search the byte sequence of the member of target structure to get the offset value. The procedure can be automatically accomplished. The example diagram is shown in Figure 8; we locate the master key parameter in the session structure.

There may be the recursive search since sometimes the byte array is referenced by a pointer or referenced by the pointer to pointer. The more layers the recursion have, the more uncertain and complicated it would be. For example, in the structure *ssl_session_st*, the buffer of the master key locates in the memory region of the current structure, but the ticket member is in the buffer referenced by the pointer *tlsext_tick* of the structure [40]. In general, when the offset information in the specific version is known, it would have few changes in other versions. Based on this assumption, the extraction of member offset of the structure can be automated to some extent.

```
Input: The set of CFGs in target binary program, CFG_SET. The set of path templates, PT
Output: The set of path similarity values, SV_SET
1: function PathSimGenerator(CFG_SET, PT)
2:   for graph in CFG_SET
3:     Initialize an array variable, stack
4:     Append first node of graph to stack
5:     while stack is not empty
6:       Pop the top item n from stack
7:       Load instruction sequence of n
8:       Get current path path_c and its label sequence label_c from stack
9:       if the length of stack is equal with length of path_c and p in PT has the same label with path_c then
10:          s_value = PATH_SIM(p, path_c)
11:          Insert s_value into SV_SET
12:          Remove n from stack
13:        else
14:          Load neighbors of n into n_nbs
15:          if b in n_nbs is not visited then
16:            if b is not in stack then
17:              Append b to stack
18:              Mark b as visited
19:            end if
20:          else
21:            Remove n from stack
22:            Mark all items in n_nbs as not visited
23:          end if
24:        end if
25:      end while
26:   end for
27:   return SV_SET
28: end function
```

Algorithm 1: Generation algorithm of path similarity.

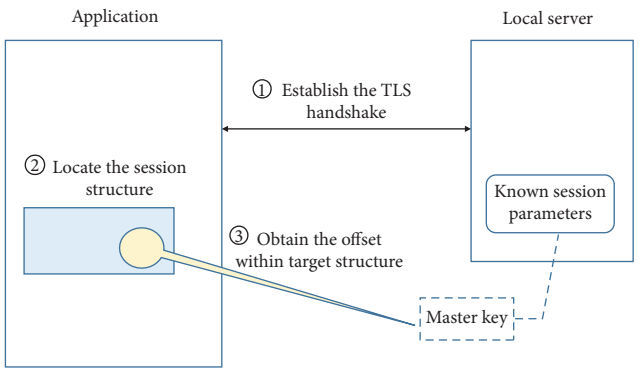


Figure 8: Locate the structure member by dynamic debugging analysis.

Besides, we should extract enough information from the target structures in order to achieve the analysis of TLS traffic. For the cipher mode of CBC, the master key or read/write key is needed, and the hash message authentication code (HMAC) is also needed. The initialization vector is the last bytes of the last cipher text which can be obtained in data packets. For the Galois counter mode (GCM) mode, the master key is enough; otherwise, another value *Salt* is needed, comparing with the CBC mode. Because *Salt* is an implicit nonce while the explicit nonce is transferred through the network; both are then combined into a single nonce value according to the TLS specification. The additional authentication data (AAD) can also be obtained from the network packet.

3.3. Data Correlation. First, all kinds of acquired data would be correlated in order to accurately decrypt the TSL traffic of the target program. Then, the unwrapped plaintext can be correlated with the files that the target process has opened, to examine if there exists the data exfiltration. The fields that data correlation depends on are shown in Figure 9, where the field plaintext represents the decrypted content of the network packet. Besides, for the case that the process information is fuzzy, the fingerprint extracted from the handshake packets can also be used to determine the crypto library loaded by the target process.

The TLS session is established upon the TLS connection; therefore, one TCP stream may contain several TLS sessions, but each data packet only corresponds to one TLS session. We deal with the packets according to the TCP stream and build the mapping to TSI for each segment of the TCP stream. As shown in Figure 10, the handshake messages are the dividing point. When the TLS handshake is finished, the decryption task of the current session will be started. Moreover, due to the session resumption mechanism, one TSI can also correspond to many TCP streams. As denoted

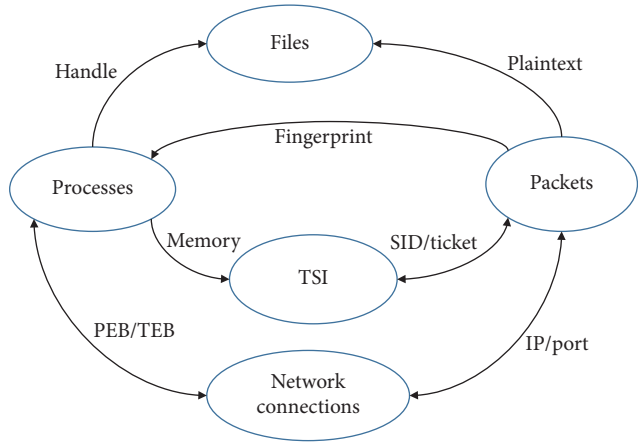


Figure 9: Correlations of multiple types of data.

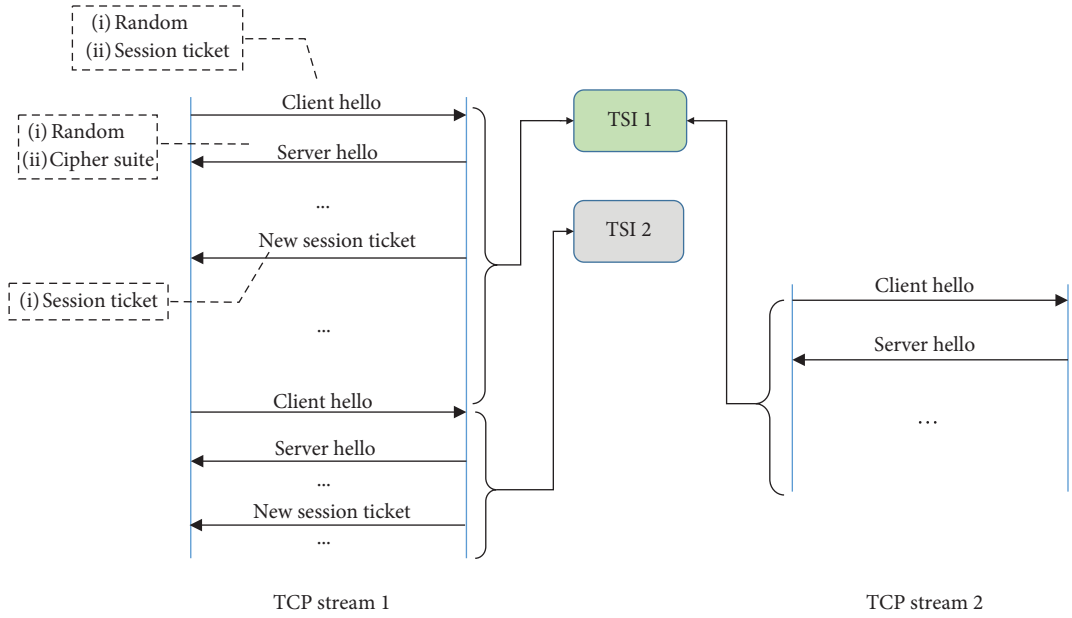


Figure 10: Correlation between TCP streams and TSIs.

in the dotted box in Figure 10, some fields would be extracted from the handshake messages, and after that the further correlation continues.

Since the TSI and network packets are obtained, respectively, and the TSI may not contain all elements needed by decryption, next we discuss how to correlate them in time. According to the RFC document, for each TLS record, the integrity needs to be verified at both ends of communication. With the cipher key, we can determine the corresponding TLS record by trying decrypting and verifying the record. At worst, we need to verify each record with all the extracted cipher keys, but the cost will rise when the number of keys increases. In the offline mode, it is still an effective method, though. In fact, we should consider the following cases.

(1) TSI contains the IP address and port. It is the perfect case because all the packets and TSI can be directly correlated by the IP and port. Of course it should be limited in a reasonable time period because the client port may be reused when the associated TCP connection is closed.

(2) TSI contains the resumption information. The basic resumption information includes session ticket and session ID, and the ticket is more widely used by servers than session ID currently. With the resumption information, the TSI can be correlated with the packet easily in most cases. The resumption information can be extracted from the TLS handshake packet *Server Hello* and *New Session Ticket* and the extension of *Client Hello*. However, there would be many connections with the same resumption information but corresponding to different TSIs. Because the session key derivation depends on the random numbers of the handshake packets, which may be updated in the new TLS session. In this case,

we should try to decrypt and verify TLS record by the set of TSIs which contains the same resumption information.

(3) TSI contains the time information. If the TSI only contains the last accessed time, meanwhile it does not contain any resumption information. In this case, we can set a time period and try to decrypt and verify the record with the TSI within this period. As is shown in Figure 11, when one record occurs, the period around the current time is chosen as the validation window. Therefore, in the short period of time, the number of TSIs used for trying is limited.

(4) TSI contains nothing except the key. Since there is no assistant data, we need to perform decryption and verification on each TLS record with all the TSIs. As is mentioned above, it will cause network delay. Therefore, when the TSI is extracted we should append the acquired time, and then the correlation will be converted to the case in the last paragraph.

4. Prototype Implementation

We implement a prototype of HTTA on Windows platform, in order to verify its feasibility. The extraction of global cache and structure member offset belong to the preliminary work before the framework running. We develop some plugins based on the *IDA Pro 6.8* tool [50]. As is shown in Figure 12, the prototype contains a kernel driver which captures the packets and extracts the TSI, an application in user mode that manages the I/O with kernel driver and decrypts the packets.

Information extraction module works as a kernel thread, which currently supports three types of browsers, *Chrome*, *Firefox*, and *Edge*. It also processes the additional encryption of Firefox. We can distinguish applications from each other by reading their process names. For *Chrome* and *Firefox*, the TSI is maintained in the parent process which has the same process name with child processes. It works on multithread mode, which creates several work threads beforehand. Multiple threads can cope with several browsers simultaneously. Moreover, TSI is stored in the render process for *Edge*, and multiple threads can improve its efficiency. We use hash table to store the extracted TSI and remove the duplicated item based on master key or send key. For the sake of performance we build several hash tables which takes port, ticket, session ID, and time as the hash key, respectively.

The packet filter module is built on the Windows Filter Platform (WFP) [51]. By taking advantage of the character of the FwpsFlowAssociateContext function that it only processes the packets of the target processes and registering the callback function in FWPM_LAYER_STREAM_V4 layer, the module can process the payload directly without maintaining the TCP sequence. All the packets are captured and cloned and then sent to the uploading buffer. Packet filter module hangs up the packets, waits for the notification, and then decides to drop it or reinject into the protocol stack.

The decryption module is implemented based on the *OpenSSL* library, which has multiple work threads and

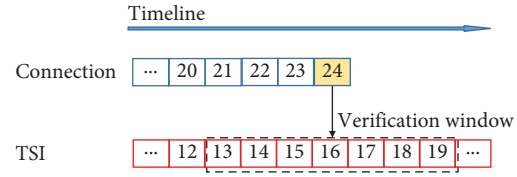


Figure 11: Verification with the TSI in a short period.

creates separate buffer queue for each TCP stream. It also parses the packet and extracts some parameters such as random number, and then derives key information based on TSI and required parameters. In practice, there is a special case that the client may send application data immediately after *Client Key Exchange* message in an initial handshake before it receives the *New Session Ticket* message from server at that moment. Hence, the sent payload cannot be correlated with TSI by the ticket. To cope with this situation, we create an additional buffer queue to decrypt them after the correlation is finished.

5. Experiments and Results Analysis

The experiments are performed on the desktop computer composed of *Intel i7-6700 @ 3.40 GHz CPU, 24 GB memory* and *Windows 10 (1607) 64-bit*. For the kernel driver testing, we also install VMware Workstation and create virtual machines with it. The configuration of virtual machine is *4 cores CPU, 8 GB memory* and *Windows 10 64-bit* operating system. Meanwhile, another virtual machine is created for a local gateway, which has 2 core CPU, 1 GB memory, and Ubuntu 16.04 64-bit operating system. The main test browsers are 64 bit *Firefox* (65.0.1), *Chrome* (72.0.3626.81), and *Edge* (38.14393.1066.0), which can represent most of the cases.

5.1. Evaluation of the TLS Session Lifecycle. First, we evaluate the lifetime of TLS sessions in the memory for different browsers in real life. The target browsers are executed with default configurations, and the network bandwidth of the experiment environment is 20 Mbps. We write a test script for automatically visiting the homepages of top HTTPS websites from Alexa, including 20 domestic and 10 foreign sites. The page will be immediately closed when it is fully loaded, and the experiment is repeated at least 10 times. Then, we count the TLS sessions by the ticket and TCP port, and also develop hook plugins for browsers to obtain the time of allocation and free for each session object. *Wireshark* is used to capture the network packets. Finally, we analyze the duration of TCP connections and correlate them with TLS sessions. Even though these websites may change dynamically and there may be errors between different tests, it will not affect the result.

In each test, about one thousand TCP connections are established in the interval of ten minutes. The result is shown in Figure 13, which is the average value based on the ten tests. It shows the lifetime distribution of TCP connections and TLS sessions in the real website access, where the left bar denotes the duration distribution of TLS sessions. Most

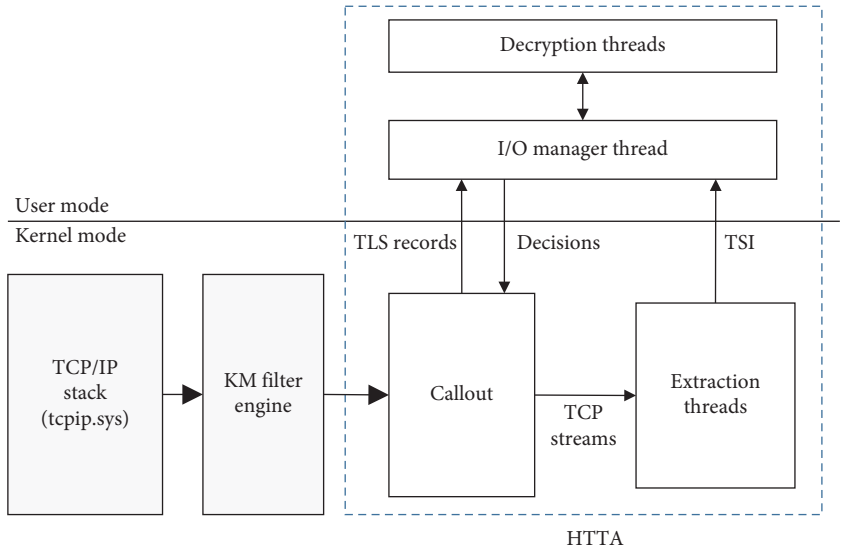


Figure 12: Prototype architecture of HTTA on Windows platform.

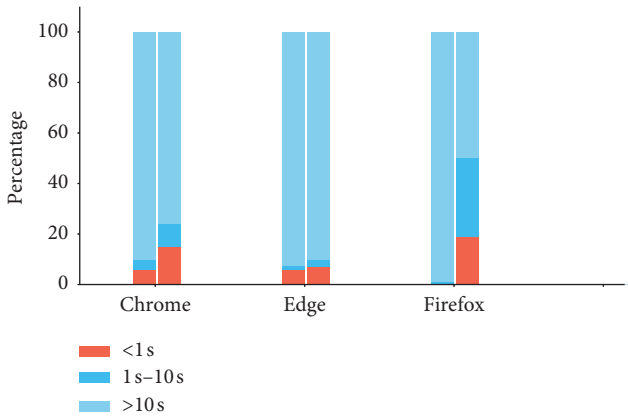


Figure 13: Duration of sessions and connections for typical browsers.

sessions can exist for a long time, so that the extraction module has enough time to obtain the TSI from the memory in the bypass mode. It also has challenges for the session that exists less than 1 s. Besides, the duration of TLS session is associated with the TCP connection in theory, which is more obvious in *Edge*. The session objects in the process memory of *IE* and *Edge* can be released in time. Other sessions with long cache time are managed by the system process. *Chrome* and *Firefox* manage sessions by themselves. Therefore, most TLS sessions exist in the memory for a long time, and the rest with short lifecycle can be released in time corresponding to the concurrent connections. As shown in the figure, in general the duration of most TLS session objects is long, especially in the case of mass data transmission, such as sending and receiving large files, so that the extraction modules have enough time to obtain them. Moreover, we find that *Chrome* do not release the session objects in time for most cases, and it may have the consideration of communication performance.

5.2. Effectiveness and Overhead Measurement. Then, we perform experiments on the real session extraction and packet decryption, in order to evaluate the effectiveness of HTTA. The extraction module extracts TSIs by traversing the cache information starting at the global variable pointer. It should be notified that the audit mode is used here, which means that we do not intercept and pend the packets; otherwise, this experiment may make no sense. We still choose three 64 bit browsers mentioned in the last section as the target programs and then install them in the virtual machine. Meanwhile, when the HTTA framework runs, the *Wireshark* and *perfmon* tool are used to capture network packets and obtain the processor time, respectively. The experiment is also repeated 10 times, while generating about one thousand connections each time, and then we calculate the average results. As shown in Figure 14, the packet decryption rates are 96.45%, 99.07%, and 95.25%, respectively. The decryption rate is over 95% for each test browser, especially near 100% for *Firefox*. Further, we investigate the reason why the decryption does not cover all the TCP connections. One is due to the statistical method, for *Chrome*, it creates multiple concurrent connections at the same time, but only one corresponding session is reserved and actually cached, which occupies about 4% in all the captured connections. For *Edge*, some connections do not come from the render process but the frame process, such as the *favicon* request which does not belong to our concerns. The rest few connections have a very short time to live, and most of that do not have the substantial data transmission, so that they are not obtained due to the very short duration of session objects in the audit mode.

In addition, as can be seen that the average CPU usage rate is low in all the tests, within 5%. Those are the counting values, while the browsers open and render the web pages. For longer period, if the idle time is considered, it would lead to a lower average processor time. The CPU occupation mainly comes from the extraction and decryption module

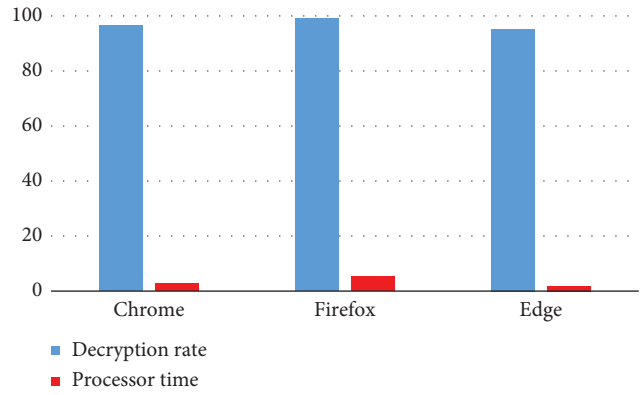


Figure 14: Decryption rate and overhead of the extraction by the global variable.

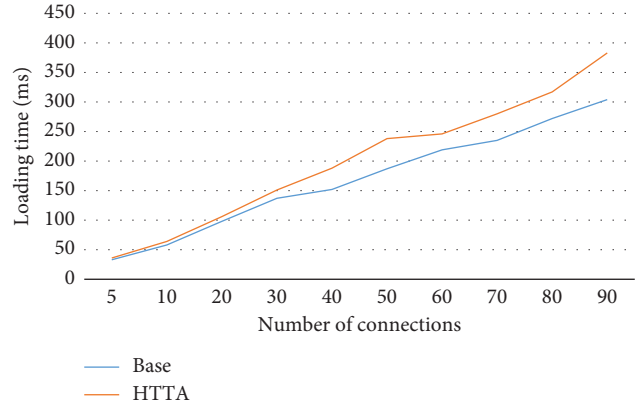


Figure 15: Overhead of the extraction by the traversal of LFH.

which are in the user space. Overhead of the packet filter module in the kernel space is negligible.

Next, we evaluate the effectiveness of the matching method based on the low fragmentation heap; consider the extreme case that all the TSIs are obtained from the heap memory by searching. As mentioned above, if the handshake packets are pending in the communication, the corresponding session information can surely be obtained in the memory space, and what needs to be concerned is the overhead and time delay. We take *Chrome* as the example and choose the *SSLClientSocketImpl* object as the target from which we can extract the TSI and TCP connection information all at once. Then, we count and observe the time delay caused by packet queuing with different number of concurrent connections in the web page, respectively. In order to avoid the impact of network transfer, a local TLS server is created. As shown in Figure 15, the page loading time does not increase obviously when the number of connections increases, and it is in the acceptable range. The main reason is that the number of concurrent connections of the browser is limited. While the number of connections grows, one traversal of the LFH chunks can obtain several target objects, and it is beneficial. Finally, the browser loads the page that generates 1000 http requests with the response size of 50 KB, and the processor time is recorded. The result shows that the processor time of entire system does not increase obviously because the extraction modules works on demand, not continuously. In fact, the search only occurs when the TSI is not obtained from the session cache.

Actually, the scenario above is common in the non-browser programs, which utilize the aforementioned TLS libraries to make secure communications. For example, some malicious programs communicate with the remote server based on the HTTPS protocol. It uses the *WinHttpSendRequest* function of the *winhttp* library to send data, which will be subsequently encrypted by functions in the *schannel* library before sending. The TSI can be easily located by magic string "Microsoft SSL Protocol Provider" in the heap memory while the handshake process is hung temporarily. So, the encrypted TLS traffic can be obtained stealthily without any preliminary analysis.

Then, we choose three structures from three browsers to evaluate the real matching effect. The structures are *sslSessionID* (0x188), *ssl_session_st* (0x198), and *CSecureSocket* (0x170), which contains the TSI information, respectively. The value in parentheses is the size of structure in the test version. In the experiment, we successively access the above-mentioned websites, the matching process continues, and the processor time is suppressed below 5%. The experimental results are shown in Table 2. There are lots of repetitive accesses to a small amount of memory blocks. After removing the duplicate addresses, the number of blocks that contain the target structures is limited. The matching process is efficient, and the time consumption can be negligible, although there are lots of nontarget blocks associated with the bucket, especially for *Edge*. One reason is that the new LFH heap is introduced on Windows 10, which is called segment heap [44], and then more memory chunks would be traversed to obtain the target structures.

5.3. Comparison with Other Methods. In the next experiments, HTTA is compared with *Fiddler* and *mitmproxy*, which are the widely used web analysis tool, to demonstrate the impact on the original communication of the method. *Fiddler* interposes a proxy to intercept the TLS connections of browsers, while *mitmproxy* is configured as the transparent mode. As is similar to the last section, we, respectively, download the page with different sizes from the local TLS server and obtain the page loading time from the development tool of the browser. As shown in Figure 16, for the transfer of small page, the page loading time is near for different methods. But there may be errors in the loading time of around 10 ms because of the counter of operating system, as shown in the interval 10 KB~100 KB. While the size of transferred packets grows, the impact on transfer delay by *Fiddler* and *mitmproxy* will increase obviously, and the impact of our method tends to be negligible. The average latency of each communication is about 5 ms, while the maximum is about 10 ms at worst. Because HTTA only needs to correlate the TSI with the connection, it only duplicates the packets and does not intercept the connection continuously. *Mitmproxy* maintains the TLS communications on both sides because of the transparent proxy; the initial handshake would cause a lot of time, and the handshake impact will be ignored while the size of packets

Table 2: Statistics of the traversal on the LFH.

Program	Number of all traversed blocks (million)	Duration	Time consumption per million blocks	Proportion of target blocks in the traversal	Number of target blocks without duplications
Firefox	338	190s	562 ms	78%	537
Chrome	326	187s	574 ms	50%	343
Edge	381	207s	543 ms	25%	363

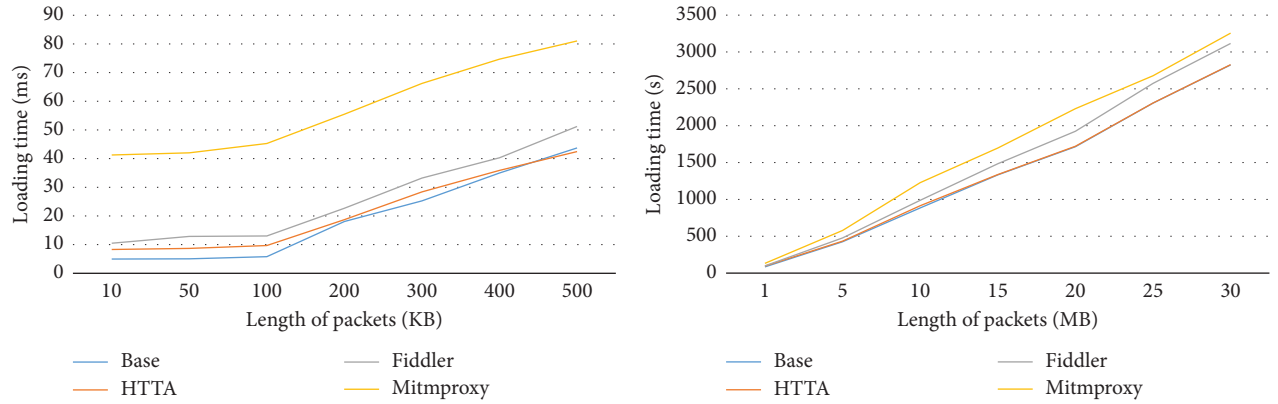


Figure 16: Comparisons with Fiddler and mitmproxy.

grows. *Fiddler* causes the lower network delay than *mitmproxy*, since it uses web proxy protocol on the client side.

Another difference is that *Fiddler* captures and processes packets in the user mode, while HTTA currently captures packets in the kernel mode and sends to the upper module. The communication between the user mode and kernel mode module may cause time delay. If HTTA works on the offline mode, it would have no impact on communications.

5.4. Correlation Cost in the Extreme Case. As aforementioned, it is easy to correlate TSI with the network connection in most cases by the connection port or session resumption information. In general, when the TLS handshake is finished, the related information can be rapidly extracted, so that the decryption can be started. The time spent of extraction is less than 5 ms. However, in the absence or loss of such information, it is necessary to try to decrypt and verify one record of TLS session to determine the relationship between the two. Specific methods are aforementioned. In this experiment, we choose some popular cipher suites of TLS protocol to evaluate and take *OpenSSL* as the crypto library even though it is not the optimal. As shown in Figure 17, with the growth of key numbers the verification time does not increase drastically for different cipher suites. If the number of key materials is less than 100, the time required to achieve the correlation with TCP stream is about 5 ms. The time consumption of the GCM mode is lower than the CBC mode. In reality, few TSI is generated at the same interval of time. Therefore, if the verification is needed, it is possible to keep the time overhead within a reasonable range. In the experiment, we use TLS 1.2 version because many websites do not support TLS 1.3. Actually, the new version only changes the handshake protocol, the symmetric encryption does not change, and it would not impact the result.

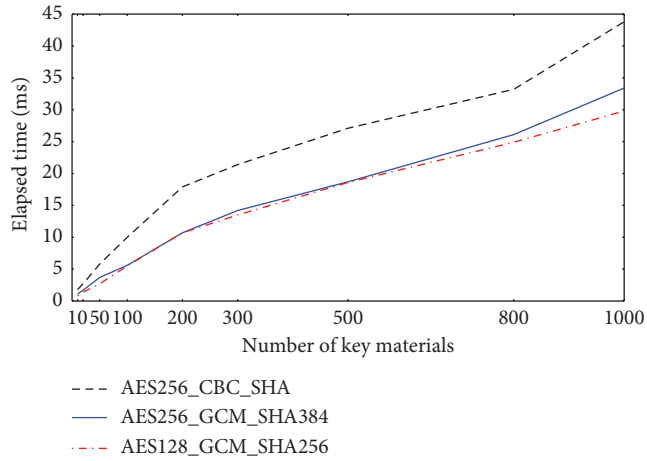


Figure 17: Elapsed time of correlation attempt for different cipher suites.

5.5. Localization of Target Objects. To support the large scale application of HTTA, we also design the helper of locating target structures and variable which can reduce the additional overhead of reverse engineering. Here, we perform the experiments to demonstrate the effect. Because there are abundant debug symbols for *Edge* and *Firefox*, we first choose *Chrome* 32-bit as the target program. To obtain the cache variable, function *SSLContext::GetInstance* is used as

the target for a test, from which the session structures can be referenced. The function in the version of *58.0.3029.110* is taken as the initial template. First, we try to get the target function by the traditional CFG matching. The base control flow graph is established, and two parameters described above, which are the matching error of the node label and the number of nodes, are also considered. Then, it matches all the CFGs in other versions of the program, so that it can quickly locate the target. The results show that it can locate the function precisely for the latter consecutive versions, while fails to find target in some prior versions. In fact, there are few subtle differences between different versions that some sides of CFG have changed. It is a challenge for applying graph isomorphism search method in that situation. Even though the complexity of graph isomorphism search is high, the actual match time is controlled because of the special constraints. Besides, we can observe that when the match error of vertex grows, the match result grows drastically. This means the fuzzy match based on graph isomorphism does not work well here.

Then, we continue to measure the ISMCP method by the next experiment, and also take the function above as the target. But we use the function in the older version *51.0.2704.103* as the template, from which we select the paths. The length of path changes from 3 to 6, and the number of each generated path locates in the range from 1 to 5. Then, the template is matched with all the functions of target version of the program by ISMCP. The result is shown in Figure 18; for each version of the program, we sort all the functions by their path similarities, from which we can see the obvious change in curvature. The target functions in different versions are in concentrated distribution, and hundreds of them rank within top 0.5%; it can still be observed clearly in the partial magnification which also bring some challenges for further analysis, although shrank the search scope. In addition, with the growth of path length, the number of generated similarity values goes down; meanwhile, the capability of locating target function also declines.

The main reason is that the CFG of the above function has a too simple structure and few vertexes. Therefore, we then choose another function *DoVerifyCertComplete*, which also changes in the different versions. Meanwhile, we can also obtain the target by one direct call reference in that function. The CFG of the function has around 20 vertexes, and structure is more complex. We choose the path length between 6 and 9, and the generated path number of each length varies from 16 to 30. The experiment steps are the same with the former one, and the result is shown in Figure 19. Now, we can see an excellent match result. For each version of the program, the top function is the target function. Most of the target functions in different versions are located within top 20, which is a better result.

Besides, the fact as shown in Figure 20, even the path number and length grows, the match time does not improve obviously. One reason is that the CFG structure is sparse, with the path length grows, the number of matched graphs will reduce, and the number of paths which can satisfy the constraint will also decrease. On the contrary, the number of matching operations increases when the path length is short. Additionally, because the module (*Chrome.dll*) that contains the target function consists of near 200 thousand functions, the process is hard to finish within the acceptable time range when the *bindiff* tool is used.

For the location of the structure members, we can use the same method. In the subsequent experiment, we choose the *SSL_SESSION_dup* function of *Chrome*, which references lots of session parameters, including master key. The version 65.0.3325.181 is used as the template to build the paths, and the path length is same as the above experiment. The result is shown as Figure 21 since the architecture of the function keeps stable in many versions; the top 10 functions are all the target function in the newer versions of target program.

Similarly, for *Firefox*, we can obtain the global cache variable by function *ssl_DestroySID*. In the next experiment, version 56.0.0 (64 bit) is used as the initial template to build the paths of ISMCP. Then, we try to locate the same function in the programs with newer version number. The result is shown in Figure 22, and the target function can be found in the top 5 values of the result for each newer version. Then, function *ssl3_FillInCachedSID* is further used in the experiment, from which we can obtain the offsets of specific session parameters. As shown in Figure 23, even though the similarities decrease in totality, the target can also be found in the top 10 of the results for version 58.0.2 and 60.0.2. But most matching values are located between the top 50 and 100 for version 65.0.1 and later, and then the difficulty of location is increased. It is better to update the matching template of the function due to the major structure change in the new versions.

Since the file size of module containing target functions is small for *Firefox*, we can perform the comparions by the *bindiff* tool. The result is not good, as shown in Figure 24. For *ssl3_FillInCachedSID,* the desired functions in the newer versions are not recognized. Function *ssl_DestroySID* is located only in version 60, due to the few changes of that version.

In fact, compared with the address change of the global variable, the member offset of target structure remains more stable, as shown in Table 3. In most cases, we only check whether the offset has changed in the new version.

5.6. Adaptation to the Program Change. The proposed method can be applied to the analysis of the popular browsers, also including other browsers with the same kernel. Meanwhile, some programs use the browser as the web component, and the corresponding traffic can also be directly analyzed, for example, *Weixin for Windows, Tencent video, Netease cloudmusic*, and so on. Besides, some client programs of cloud disk use *wininet* library for the secure communications, which can also be directly analyzed. Furthermore, the address of the target global variable may change in different versions of the program due to the compilation. So, we need to construct the offset database of the target variable in advance. For the structure member, it generally changes when the version of program has the major change.

Moreover, we investigate the variation of the target object size, which is used in the experiment of traversing the LFH. As shown in Table 4, the size can remain stable in different versions, which is sensitive to the major version

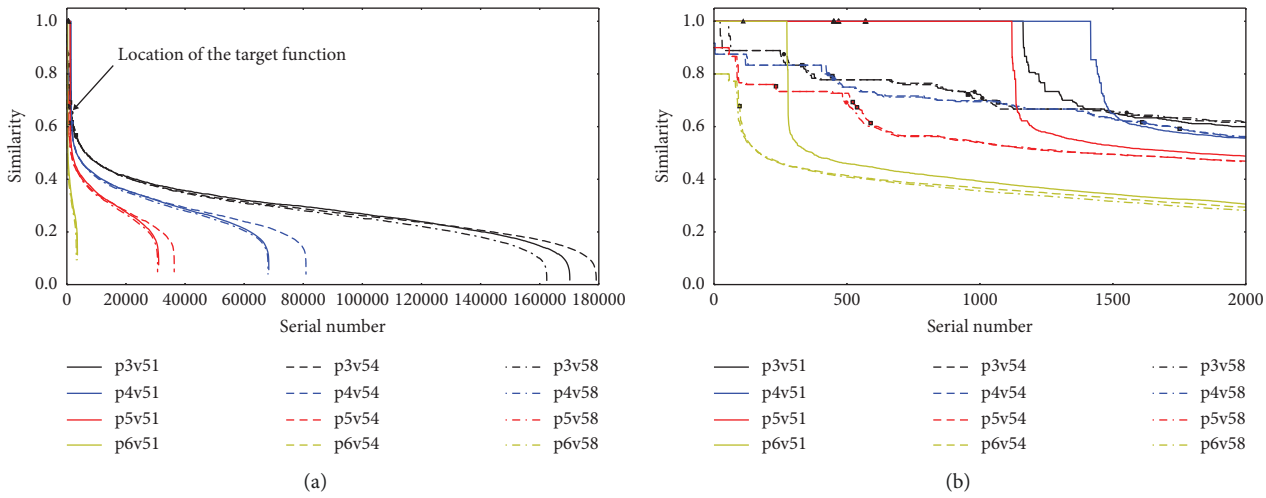


Figure 18: Result 1 of Chrome based on ISMCP. P3 denotes that the path length is 3, v51 denotes that the major version is 51, and others are similar.

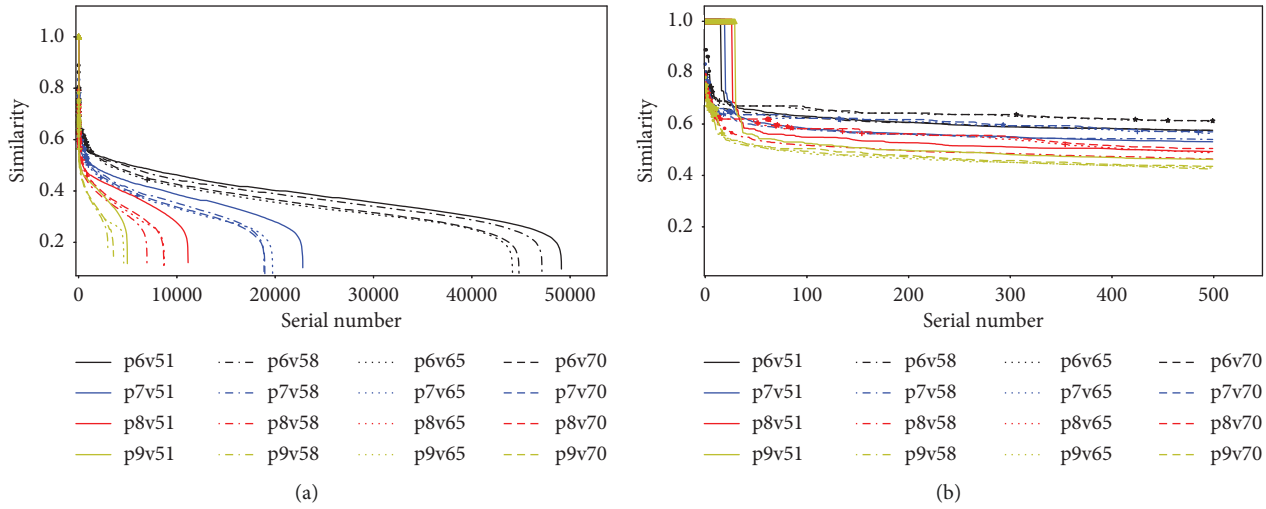


Figure 19: Result 2 of Chrome based on ISMCP.

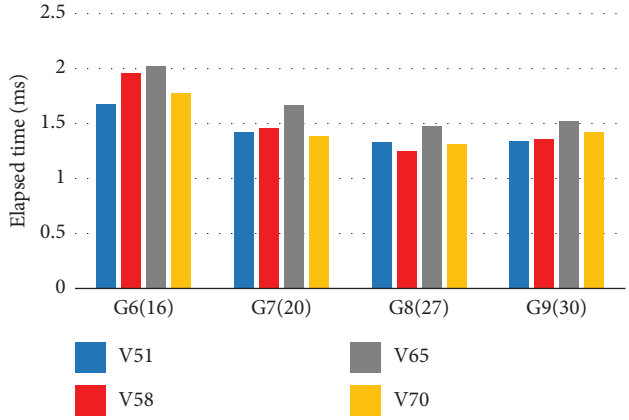


Figure 20: Execution time of matching per graph in different conditions. G6 (16) denotes that the path length is 6 and the number of paths is 16, others are similar.

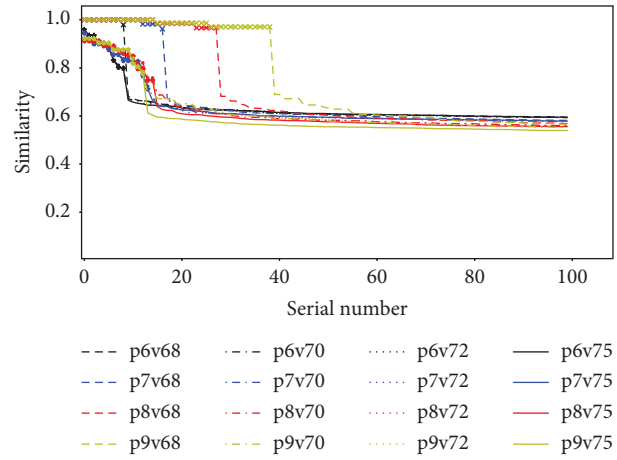


Figure 21: Result 3 of *Chrome* based on ISMCP.

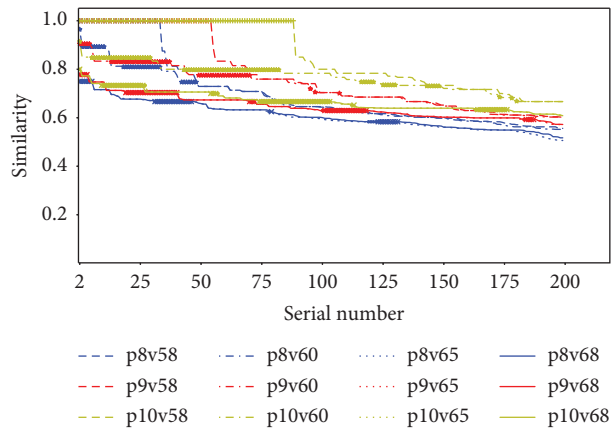


Figure 22: Result 1 of *Firefox* based on ISMCP.

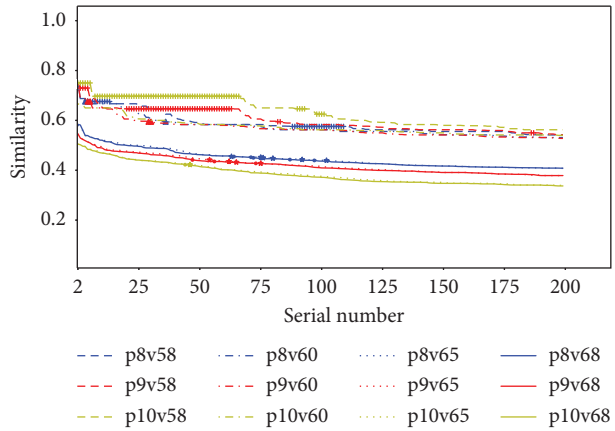


Figure 23: Result 2 of Firefox based on ISMCP.

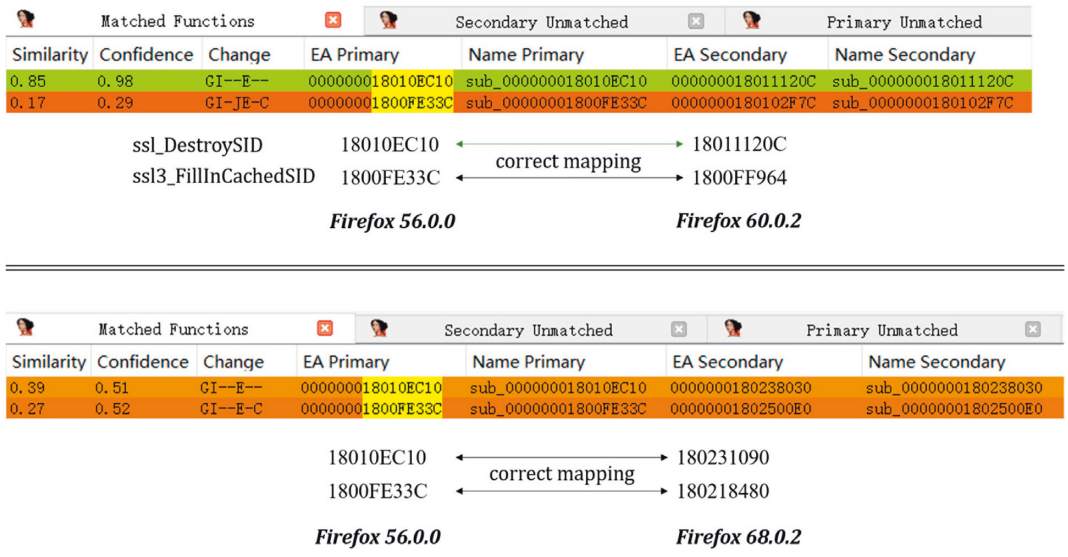


Figure 24: Comparisons of different *Firefox* versions by bindiff.

Table 3: Variation of the parameter offset in different versions.

Program	Structure	Member	Version	Offset value
Firefox	sslSessionID	keys	56.0	0xAC
			60.0.2	0xB5
			65.0.1	0xB5
Chrome	ssl_session_st	master_key	65.0.3325.181	0xC8
			70.0.3538.77	0xC8
			75.0.3770.80	0xC8
Edge	CSslUserContext	ReadWriteKey	10.0.14393.1613	0x18
			10.0.16299.15	0x18
			10.0.17134.1	0x18

Table 4: Variation of target objects of different versions.

Program	Object	Version	Size (bytes)
Firefox	sslSessionID	56.0	0x190
		60.0.2	0x188
		62.0.3	0x188
		65.0.1	0x188
Chrome	ssl_session_st	61.0.3163.100	0x1B0
		65.0.3325.146	0x198
		70.0.3538.77	0x198
		72.0.3626.121	0x198
IE/Edge	CSecureSocket	11.0.14393.1770	0x170
		11.0.14393.2273	0x170
		11.0.16299.15	0x180
		11.0.17134.1	0x198

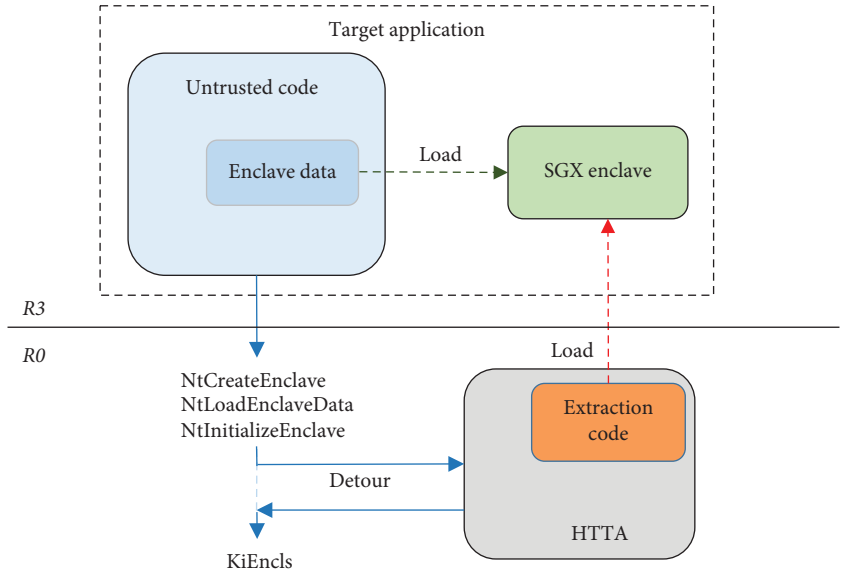


Figure 25: A solution to the case of SGX.

changes. In practice, we can properly increase the search range to improve the adaptability.

6. Discussion

The proposed method is practical and universal, as it does not depend on vulnerabilities of the program and protocol. It is mainly applicable for browsers and analogous programs since we discover the uniform pattern of information extraction, and it is applicable for the programs which adopt the similar crypto infrastructure. But it is difficult to directly extract the session information for the program that has the personalized implementation of TLS protocol. In the situation, the individual reverse engineering work is needed to make the analysis cover the program.

The version of browser program may change frequently; in most cases, we only need to check the variation with low cost, due to the semiautomated method. Little manual work

is needed to ensure the accuracy in practice. But when the target function in the new version has major changes, we would spend much time to update the new function template to locate the target parameters.

The noninvasive method of this paper means that it does not intervene the execution flow of target program. As mentioned above, the solution has the interception mode and audit mode. In the interception mode, few related handshake packets will be intercepted at the system kernel level, but there is no reassembly and decryption. Meanwhile, in the audit mode, the decryption rate may not reach the one hundred percent, but it can minimize the impact on the target system and application.

Besides, as a supplement, we also propose the extraction method based on the low fragmentation heap. When the amount of traffic of the program is small, the corresponding heap bucket may not be activated. If the bucket is needed to be forcedly activated, there would be the thread injection which slightly interferes with the target system.

For the standalone deployment, other security products may exist. Because the solution does not involve the function hooking methods, they can coexist. Moreover, as a defense solution, we can also add the related modules to the while list of security products.

While the Intel SGX is also a barrier to the solution, it has not been used by target programs. In that case, it needs to load the extraction code into the same enclave of the process when the target program initializes, which also causes the intervention. Because as a defense solution the module of HTTA can start in advance of other applications. One response solution on the Windows platform is shown in Figure 25. When the target program calls the *NtInitializeEnclave* function that corresponds to the *EINIT* privileged instruction, we load the enclave part of extraction module to the same enclave. In this case, the kernel function hooking is needed, and there are many stable methods.

7. Conclusions

In this paper, we propose a new TLS traffic large-scale automated analysis method for browsers and analogous programs, which is efficient and transparent to the programs. It extracts multiple types of data by several modes and correlates them together to accomplish the overall analysis of the target in real time. The experimental results have shown that the method can effectively capture and analyze all the packets, with the high decryption rate and low runtime overhead. Moreover, we propose the aided location method of targets to solve the program diversity problem, which can reduce the workload of binary analysis and support the framework.

In this paper, we mainly focus on the researching and performing experiments on the browser programs and would pay more attentions to other programs in the near future. Moreover, the automated method of binary reversing and parameter extraction should be continuously improved.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (General Program) under grant no. 61572253 and the Innovation Program for Graduate Students of Jiangsu Province, China (grant no. KYLX16_0384).

References

[1] G. K. Jayasinghe, J. S. Culpepper, and P. Bertok, "Efficient and effective realtime prediction of drive-by download attacks," *Journal of Network and Computer Applications*, vol. 38, no. 1, pp. 135–149, 2014.

[2] "RFC6101," August 2018, https://tools.ietf.org/html/rfc6101.

[3] "RFC5246," August 2018, https://tools.ietf.org/html/rfc5246.

[4] "RFC2818," August 2018, https://tools.ietf.org/html/rfc2818.

[5] "CVE-2014-0160," August 2018, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160.

[6] S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage, "When private keys are public: results from the 2008 Debian OpenSSL vulnerability," in *Proceedings of the ACM SIGCOMM Conference on Internet Measurement*, pp. 15–27, ACM, Chicago, Illinois, USA, November 2009.

[7] "CVE-2012-4929," August 2018, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-4929.

[8] T. Duong and J. Rizzo, "Here come the ⊕ ninjas," August 2018, http://www.hpcc.ecs.soton.ac.uk/~dan/talks/bullrun/Beast.pdf.

[9] N. J. A Fardan and K. G. Paterson, "Lucky thirteen: breaking the TLS and DTLS record protocols," in *Proceedings of the IEEE Symposium on Security and Privacy 2013*, pp. 526–540, IEEE, San Francisco, CA, USA, May 2013.

[10] M. Stevens, A. Sotirov, J. Appelbaum, A. Lenstra, D. Molnar, and D. A. Osvik, "Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate," in *Advances in Cryptology*, pp. 55–69, Springer, Berlin, Germany, 2009.

[11] K. Dan, M. L. Patterson, and L. Sassaman, "PKI layer cake: new collision attacks against the Global X.509 infrastructure," in *Proceedings of the International Conference on Financial Cryptography and Data Security*, pp. 289–303, Springer-Verlag, Tenerife, Spain, January 2010.

[12] D. Akhawe and A. P. Felt, "Alice in warning land: a large-scale field study of browser security warning effectiveness," in *Proceedings of the USENIX Conference on Security 2013*, pp. 257–272, USENIX Association, Washington, DC, USA, August 2013.

[13] A. Parsovs, "Practical issues with TLS client certificate authentication," in *Proceedings of the Network and Distributed System Security Symposium 2014*, San Diego, CA, USA, February 2014.

[14] M. Kranch and J. Bonneau, "Upgrading HTTPS in mid-air: an empirical study of strict transport security and key pinning," in *Proceedings of the Network and Distributed System Security Symposium*, San Diego, CA., USA, February 2015.

[15] "TLS 1.3 draft," August 2018, https://tools.ietf.org/html/draft-ietf-tls-tls13-28.

[16] T. Nichols, A. Bates, J. Pletcher et al., "Securing SSL certificate verification through dynamic linking," in *Proceedings of the ACM Sigsac Conference on Computer and Communications*

Security, pp. 394–405, ACM, Scottsdale, AZ, USA, November 2014.

[17] "HTTP strict transport security (HSTS)," August 2018, https://tools.ietf.org/html/rfc6797.

[18] "Public key pinning extension for HTTP," August 2018, https://tools.ietf.org/html/rfc7469.

[19] "Certificate transparency," August 2018, https://tools.ietf.org/html/rfc6962.

[20] M. Husák, M. Čermák, T. Jirsík, and P. Čeleda, "HTTPS traffic analysis and client identification using passive SSL/TLS fingerprinting," *EURASIP Journal on Information Security*, vol. 2016, no. 1, 2016.

[21] "Deciphering malware's use of TLS (without Decryption)," August 2018, https://arxiv.org/pdf/1607.01639.pdf.

[22] X. C. Carnavalet and M. Mannan, "Killed by proxy: analyzing client-end TLS interception software," in *Proceedings of the the Network and Distributed System Security Symposium*, San Diego, CA, USA, February 2016.

[23] Z. Durumeric, Z. Ma, D. Springall et al., "The security impact of HTTPS interception," in *Proceedings of the the Network and Distributed System Security Symposium*, San Diego, California, USA, February-March 2017.

[24] "Fiddler," August 2018, http://www.telerik.com/fiddler.

[25] "Mitmproxy," August 2018, https://github.com/mitmproxy/mitmproxy.

[26] "NSS key log format," August 2018, https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS/Key_Log_Format.

[27] "Jkambic," August 2018, http://media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/DEFCON-24-Jkambic-Cunning-With-Cng-Soliciting-Secrets-From-Schannel-WP.pdf.

[28] J. Rizzo and T. Duong, "Practical padding oracle attacks," in *Proceedings of the 4th USENIX Workshop on Offensive Technologies*, pp. 1–8, USENIX Association, Washington, DC, USA, August 2010.

[29] "BREACH," August 2018, https://github.com/nealharris/BREACH.

[30] M. Marlinspike, "More tricks for defeating SSL in practice," in *Proceedings of the DEFCON'17*, Winchester, NV, USA, July-August 2009.

[31] J. Liang, J. Jiang, and H. Duan, "When HTTPS meets CDN: a case of authentication in delegated service," in *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, pp. 67–82, IEEE, San Jose, CA, USA, May 2014.

[32] Y. Jia, Y. Chen, X. Dong, P. Saxena, J. Mao, and Z. Liang, "Man-in-the-browser-cache: persisting HTTPS attacks via browser cache poisoning," *Computers and Security*, vol. 55, pp. 62–80, 2015.

[33] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, "BlindBox: Deep packet inspection over encrypted traffic," *Acm Sigcomm Computer Communication Review*, vol. 45, no. 4, pp. 213–226, 2015.

[34] L. Wu, B. Chen, S. Zeadally, and D. He, "An efficient and secure searchable public key encryption scheme with privacy protection for cloud storage," *Soft Computing*, vol. 22, no. 23, pp. 7685–7696, 2018.

[35] B. Dolan-Gavitt, T. Leek, J. Hodosh, and W. Lee, "Tappan Zee (north) bridge: mining memory accesses for introspection," in *Proceedings of the ACM Sigsac Conference on Computer & Communications Security*, pp. 839–850, ACM, Berlin, Germany, November 2013.

[36] B. Taubmann, C. Frädrich, D. Dusold, and H. P. Reiser, "TLSkex: harnessing virtual machine introspection for decrypting TLS communication," *Digital Investigation*, vol. 16, pp. S114–S123, 2016.

[37] Q. Feng, A. Prakash, M. Wang, C. Carmony, and H. Yin, "ORIGEN: automatic extraction of offset-revealing instructions for cross-version memory analysis," in *Proceedings of the ACM on Asia Conference on Computer and Communications Security*, pp. 11–22, ACM, Xi'an, China, May-June 2016.

[38] P. Vadrevu, J. Liu, B. Li, B. Rahbarinia, K. H. Lee, and R. Perdisci, "Enabling reconstruction of attacks on users via efficient browsing snapshots," in *Proceedings of the Network and Distributed System Security Symposium 2017*, San Diego, CA, USA, February-March 2017.

[39] "Desktop browser market share worldwide," August 2018, http://gs.statcounter.com/browser-market-share/desktop/worldwide.

[40] "Chromium source code," August 2018, https://chromium.googlesource.com/chromium/src.git/+/62.0.3188.1/net/ssl/.

[41] "OpenSSL," August 2018, https://www.openssl.org.

[42] "LibreSSL," August 2018, http://www.libressl.org.

[43] "Firefox source code," August 2018, http://releases.mozilla.org/pub/firefox/releases/.

[44] "Segment heap internals," August 2018, https://www.blackhat.com/docs/us-16/materials/us-16-Yason-Windows-10-Segment-Heap-Internals.pdf.

[45] F. E. Allen, "Control flow analysis," *Acm Sigplan Notices*, vol. 5, no. 7, pp. 1–19, 1970.

[46] L. Luo, J. Ming, D. Wu, P. Liu, and S. Zhu, "Semantics-based obfuscation-resilient binary code similarity comparison with applications to software plagiarism detection," in *Proceedings of the ACM Sigsoft International Symposium on Foundations of Software Engineering*, pp. 389–400, ACM, Hong Kong, China, November 2014.

[47] S. Alam, R. N. Horspool, and I. Traore, "A framework for metamorphic malware analysis and real-time detection," *Computers and Security*, vol. 48, pp. 212–233, 2015.

[48] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub) graph isomorphism algorithm for matching large graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 10, pp. 1367–1372, 2004.

[49] G. Dan, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge University Press, London, UK, 1997.

[50] "IDA," August 2018, https://www.hex-rays.com/products/ida/index.shtml.

[51] "Windows filtering platform," August 2018, https://docs.microsoft.com/zh-cn/windows/desktop/FWP/windows-filtering-platform-start-page.

Hindawi
Security and Communication Networks
Volume 2019, Article ID 3648671, 14 pages
https://doi.org/10.1155/2019/3648671

Research Article

A Bitwise Design and Implementation for Privacy-Preserving Data Mining: From Atomic Operations to Advanced Algorithms

Baek Kyung Song, Joon Soo Yoo, Miyeon Hong, and Ji Won Yoon

Korea University, Seoul, Republic of Korea

Correspondence should be addressed to Ji Won Yoon; jiwon_yoon@korea.ac.kr

Received 7 April 2019; Accepted 14 August 2019; Published 16 October 2019

Academic Editor: Stelvio Cimato

Copyright © 2019 Baek Kyung Song et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Homomorphic encryption (HE) is considered as one of the most powerful solutions to securely protect clients' data from malicious users and even severs in the cloud computing. However, though it is known that HE can protect the data in theory, it has not been well utilized because many operations of HE are too slow, especially multiplication. In addition, existing data mining research studies using encrypted data focus on implementing only specific algorithms without addressing the fundamental problem of HE. In this paper, we propose a fundamental design and implementation of data mining algorithm through logical gates. In order to do this, we design various logic of atomic operations in encrypted domain and finally apply these logic to well-known data mining algorithms. We also analyze the execution time of atomic and advanced algorithms.

1. Introduction

With the progress of storage in the cloud server, advanced data process and analysis using machine learning and data mining techniques are developed to extract valuable information. However, the concern about the data privacy and security issues has occurred in storing and managing information in cloud servers. This is because the server must decrypt the data in order to process the data encrypted in conventional cryptosystems such as AES and DES, even though the client transmits the data to the server in encrypted form. Eventually, users must share the decryption key with the cloud, which can lead to data infringement by a malicious server.

Homomorphic encryption (HE) [1, 2] is mentioned as one of the most powerful solutions to the data security problem in the cloud, since the data can be processed in the encrypted domain without decryption. However, data analysis with HE is not so popular in real world although it is highly recommended for providing the proper security to the cloud. The major reason is the fact that it is difficult to link HE and machine learning. As known, HE is a new cryptosystem which uses profound, mathematical property with lattice, which makes it difficult for the data scientists to understand and use.

In addition, a few well-known HE algorithms support only very simple operations such as addition and multiplication between integers. Although Gentry [3] presented fully homomorphic encryption (FHE) which allows all operations on the ciphertext to be theoretically unlimited, it had many limitations in adapting to the real cloud model [4]. Since the implementation and development of the encryption algorithm are not main interest to theoretical cryptographers, the practical usage and implementation are rarely developed compared to the theoretical progress in FHE. Therefore, to date, FHE has been limited to be applied only to specific algorithms without solving the fundamental problems of FHE [5–11].

From this point of view, we propose a FHE computation method that can be applied more generally by using bitwise logical circuits, rather than algorithms that operate only under certain conditions. By designing the basic operations necessary for machine learning, we make a universal link between HE and machine learning. People who are studying FHE can easily apply machine learning with homomorphic operations. Furthermore, machine learning researchers will

be able to run data-driven data analysis algorithms with encrypted data although they do not have the knowledge about FHE at all.

Our contribution of this paper is threefold:

(i) In order to build simple data mining techniques with FHE, we design various atomic operations including absolute value operation, multiplication, comparison, and sorting through the gate operation provided by the TFHE library

(ii) In contrast to the integer-based FHE scheme in which possible operations are limited, all the operations including division and log can be designed in the bit-based FHE scheme

(iii) We finally demonstrate the applicability of the several well-known data mining techniques using our proposed bitwise FHE schemes: the linear regression, the logistic regression, k-NN classifier, and k-means clustering

2. Background

2.1. Homomorphic Encryption. Homomorphic encryption (HE) [1, 2] is a cryptosystem in which the result of operations between ciphertexts is equal to the result of operations between plaintexts when decrypted. The operations on the ciphertexts of a and b can be expressed as $a \circ b = \mathbb{D}[\mathbb{E}[a] \bullet \mathbb{E}[b]]$ where $\mathbb{E}[\cdot]$ and $\mathbb{D}[\cdot]$ denote encryption and decryption, respectively.

The concept of HE was first presented in 1978 by Rivest et al. [12]. Many HE schemes have been introduced since then, and the most popular one was the Paillier cryptosystem, proposed by Paillier [13] in 1999. However, they were partial HE with a limited number of operations since the encryption noise is amplified each time the operation is performed. The solution to this noise accumulation problem was the fully homomorphic encryption (FHE) of Gentry [3] in 2009. Gentry [3] proposed a bootstrapping algorithm that removes accumulated noise, thereby eliminating the limit on the number of operations. However, this Gentry [3] technique had to encrypt each plaintext bit by bit. It was a heavy burden on memory because the size of the ciphertext was so large. In addition, the bootstrapping operation was performed with a very complicated algorithm, so it took dozens of minutes to bootstrap a bit. For these reasons, many FHE libraries now use integer-based schemes, but this also has the disadvantage that the possible operations are very limited.

2.1.1. TFHE Library for FHE. In 2017, Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène proposed TFHE [14] library which is an improved version of FHEW [15] library. It has the bit-by-bit encryption scheme similar to Gentry's initial FHE [3]. However, unlike [3], TFHE has constructed operations in a more fundamental way than addition and multiplication between ciphertexts. It is the binary circuit that was used for the encrypted bits operation. In other words, TFHE supports NOT, AND, OR, NAND, NOR, XOR, and XNOR gate operations between encrypted bits, allowing users to construct encrypted circuits using these logical operations. Another advantage of TFHE is that it efficiently solves the bootstrapping problem, which was the biggest obstacle to using FHE. This is designed to perform a bootstrapping function automatically whenever a single operation is performed, unlike the conventional FHE, in which a direct bootstrapping must be performed to remove noise each time a certain number of operations are performed. In other words, it is possible to perform computation without limitations. Here, the bootstrapping algorithm is performed with a time of less than one 0.1 second and has the fastest performance among all of the preceding FHE schemes.

In addition, through supporting the multiplexer function, convenience of implementation and speed of circuit are more improved. In the below function, a is a multiplexer factor and outputs either b or c depending on the value:

$$\mathrm{MUX}(a, b, c) = \begin{cases} b, & (\text{if } a = 0), \\ c, & (\text{if } a = 1). \end{cases} \tag{1}$$

2.2. Data Mining and Machine Learning Algorithms

2.2.1. Linear Regression. Linear regression is the most popular model for predicting target value of y. It is the method that estimates the coefficients of the linear equation, involving one or more independent variables. Several types of process exist to optimize the values of the coefficients. We focus on gradient descent, iteratively minimizing the error of the training data.

2.2.2. Logistic Regression. Logistic regression is a special case of generalized linear model in which the target variable is binary such as pass or fail, live or death, etc. In general, logistic regression makes an inference on parameters of sigmoid function which determines classification of modeling binary or categorical dependent variables.

2.2.3. k-Nearest Neighbors (kNN) Classification. In data mining, the k-nearest neighbors algorithm is one of the most well-known and useful supervised methods for classifying a dataset. Given the classified data with several classes, the kNN determines the class of new input data based on its neighbors. At this time, the label of the input data is set to the largest number of labels of the closest k data. In addition, there are many ways to calculate the distance between data, typically Euclidean distance. Depending on which distance measurement method is used, different results may be obtained.

2.2.4. k-Means Clustering. Unlike kNN, the k-means clustering algorithm grasps the relationship of unlabeled data and clusters them into k clusters. The k-means clustering sets the representative value of each cluster and assigns each data to the cluster with the closest representative value. After forming clusters for k representative values initially set arbitrarily, the mean of each cluster is newly representative of

each cluster. This process is repeated until the cluster converges, and finally, the data are clustered into k clusters. The result of the k-means is affected by the distance measurement method as well as the kNN.

3. Problems

3.1. FHE for Machine Learning. Although machine learning and FHE have long history, their research has been conducted separately for a long time. Recently, as the era of cloud computing comes, privacy-preserving machine learning and data mining have been introduced as a hot topic. There have been several studies on connecting FHE and machine learning [6, 7, 9, 10, 16].

However, as mentioned in Section 2, there is a limitation that it is difficult to apply FHE to machine learning algorithms because it is only possible to perform limited operations such as addition and multiplication in most libraries. Accordingly, existing machine learning studies using encrypted data have focused on implementing specific algorithms such as Naïve Bayes classifier [9] or linear regression [16]. In addition, since FHE requires complex theoretical knowledge, it is difficult for general machine learning engineers to understand its concept. Worse, in order to use the FHE scheme, we need a technique to replace all operations on plaintext with homomorphic operations.

In this paper, we focused on how to efficiently implement basic atomic operations and universal application to various machine learning algorithms. These studies will be a good mediator between FHE and machine learning.

3.2. Integer-Level Encryption vs. Bitwise Encryption. The FHE, which operates in integer space, takes scalar integers or polynomials with integer coefficients as input and then performs an operation on an integer basis. Therefore, additional integer encoding is required for real data that are not integers. Previous research studies about FHE application have used the rounding function to convert real numbers to integers for the encoding and decoding processes. Most of them used the scaling constant k before rounding to preserve the original number. In order to recover the encoded value, k must be divided from the decrypted result as follows:

(1) Encoding: $\lfloor k \times a \rceil$ for a plaintext $a \in \mathbb{R}$

(2) Encryption: $c = \mathbb{E}[\lfloor k \times a \rceil]$

(3) Decryption and decoding: $a \approx (1/k) \cdot D[c]$

However, there is a problem with this method, which is to use an approximation rather than an accurate data. The approximation accuracy of the data is determined by the scaling constant, and the user must also determine this constant.

On the other hand, bitwise encryption does not require encoding process to an integer because all real-valued data can be represented in bits. In addition, since the computer stores and processes data on a bit-by-bit basis, a generalized encryption scheme can be easily applied to any data.

In this paper, we introduce the logic of various operations for the bitwise encryption scheme using the TFHE library. We present a method for constructing atomic operations using the circuit operation for each bit after converting integer data into bits. Table 1 shows the logical operators used in this study and their notation.

4. Designing Homomorphic Atomic Operations

Our method uses the TFHE library, so we perform all operations on a bit-by-bit basis. This is similar to the way that binary data in a plaintext are processed by a computer using AND/OR/NAND/NOR/XOR/XNOR/NOT gates. However, since we do not know actual values to be computed, the operations should be differently designed from algorithms in the plaintext, such as using ciphertext in if-statement (for example, "If ciphertext = $\mathbb{E}[0]$, then follow below command"; in this case, we cannot compare ciphertext and $\mathbb{E}[0]$ typically). Considering these characteristics, we introduce a new design of the atomic operations in this section. The atomic operations include Addition, 2's Complement, Subtraction, Equivalent Comparison, Large and Small Comparison, Shift, Absolute, Multiplication, and Division. Note that Addition, Subtraction, and Multiplication among these atomic operations have already been introduced in the literature [14, 17]. However, other atomic operations have rarely been studied although they are highly significant for numerical computation. We demonstrate the description and algorithms of both already and rarely studied atomic operations in this section because they are separately classified as homomorphic atomic operations from the advanced homomorphic data mining algorithms in Section 4.

All algorithms introduced in this paper are implemented and evaluated with Intel i7-7700 3.60 GHz, 8.0 GB RAM, and Ubuntu 16.04.4 LTS.

4.1. Addition Operation. Addition is one of the most basic operations. There are many ways to implement full adder circuit with basic gates such as 9 NAND gates and 7 NOR and 5 NOT gates. However, since the number of basic gates is relative to speed of the circuit in the TFHE library, addition can be more efficiently designed by using only 2 XOR, 2 AND, and 1 OR gates. More details are described in Figure 1.

In the circuit diagram of Figure 1, the least significant bit (lsb) of a and b is input to the upper bit input, and c_0, which is the lsb of the carry, is initialized to $\mathbb{E}[0]$. s_i passing through the circuit is the sum of the corresponding bits, and c_{i+1} is the carry of the next bit.

4.2. 2's Complement Operation. It is necessary to express a negative number in order to perform an integer binary data operation. There are two ways to represent negative numbers in a computer, mainly the 1's complement method and 2's complement method. The 1's complement method has a simpler advantage than the 2's complement method when representing a negative number. The desired number is operated through a XOR gate with a single bit 1. The process can be replaced to taking the NOT gate for every bit of the desired number. This is because the NOT gate is significantly faster than the XOR gate. However, the 1's complement

Table 1: The notations of the logical operators.

Operator	NOT	AND	OR	NAND	NOR	XOR	XNOR
Notation		$\wedge$	$\vee$	$\overline{\wedge}$	$\overline{\vee}$	$\oplus$	$\odot$

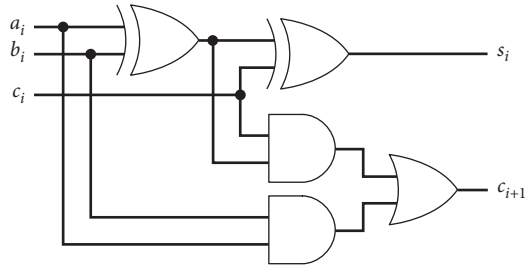


Figure 1: The circuit design for the binary full adder in the FHE scheme.

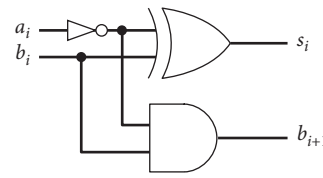


Figure 2: The circuit design for 2's complement in the FHE scheme.

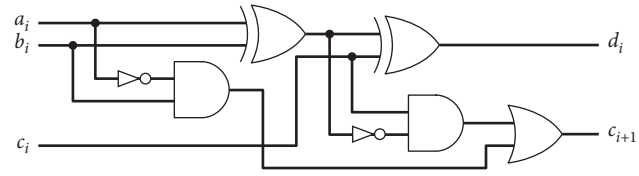


Figure 3: The circuit design for subtraction.

method has two ways of representing 0, and it is necessary to use a logic different from the plaintext to perform operations such as addition and subtraction. The method of improving this is the 2's complement method, which is represented by adding the integer 1 in the 1's complement method. Since, when 1 is represented by a binary number, it is filled with zeros except for lsb, the carry can be added to the next bit of 1 to perform addition. Therefore, when adding, it is possible to reduce the speed by adding the NOT gate to the half adder which does not need carry, without using the previous full adder: $b_{i+1} = a_i \wedge b_i$ and $s_i = a_i \oplus b_i$. This is expressed by a circuit as shown in Figure 2.

Set the number a and $b = [00 \ldots 01]$ to take the 2's complement operation and input from each lsb. The output s_i from the above circuit is the result of the corresponding bits; the carry is b_{i+1}, which is the next input.

4.3. Subtraction Operation. In a typical computer environment, you can implement subtraction using the 2's complement method and addition, so subtraction logic is not implemented separately. However, subtraction can be processed using the 2's complement method and addition as in plaintext, but it can be newly implemented with 2 XOR, 2 AND, 1 OR, and 2 NOT gates. The detailed circuit diagrams are demonstrated in Figure 3.

Subtraction enters the input from lsb of a and b. d_i passed through the circuit is the result of the subtraction of that bit, and c_{i+1} is the carry of the next bit. d_i and c_{i+1} are defined according to their value after defining D as shown in the following equation: $D = a_i - b_i - c_i$. In this equation, if $D = 1$, then $d_i = 1$ and $c_{i+1} = 0$. If $D = 0$, then $d_i = 0$ and $c_{i+1} = 0$. If $D = -1$, then $d_i = 1$ and $c_{i+1} = 1$. And if $D = -2$, then $d_i = 0$ and $c_{i+1} = 1$.

4.4. Comparison Operation

4.4.1. Equivalent Comparison. Equivalent comparison in plaintext compares each bit for two input values and outputs 1 if all are equal and 0 if there are other values. However, in encrypted data, it is possible to determine whether each bit is the same through an XOR gate, but since it comes out encrypted, it does not know what the value is. Therefore, to get the results we want, all the results of the XNOR gate of each bit are operated with the AND gate as shown in Figure 4. Then, $\mathbb{E}[0]$ is output when there are different bits in two inputs, and $\mathbb{E}[1]$ is output if each bit is the same value. Then, if the input values are different, $\mathbb{E}[0]$ is returned for the output and $\mathbb{E}[1]$ for the same input values.

4.4.2. Large and Small Comparison. We will explain this as a large comparison because the large comparison and the small comparison are logically similar. In a computer, large comparison is a system that outputs results when bits with different values are compared while comparing from upper bit to lower bit. However, since it is not known whether the value of the comparison of each bit is ciphertext of 1 or ciphertext of 0, it does not know which bit has a different value and which of the two numbers is larger. Thus, we have to use the new logic.

First, let us consider the sign bit of the result of subtracting the preceding number from the latter number of two inputs. If the preceding number is less than or equal to the latter number, $\mathbb{E}[0]$ is output and larger $\mathbb{E}[1]$ is output. Therefore, we will use this subtraction to make a large comparison. However, considering the speed of the circuit, we will use a method that uses a multiplexer function and XNOR gate. The detailed circuit diagrams are demonstrated in Figure 5. The result of the comparison is the result of repeating the circuit by the length of the data.

Larger than or equivalent comparison or smaller than or equivalent comparison can take a NOT gate as the result of a small comparison or a large comparison, respectively.

4.5. Shift Operation. Since the ciphertext is encrypted bitwise, it can be shifted in the same way as for the shift in plaintext. Shift the k bits to the left and fill the empty right k bits with $\mathbb{E}[0]$. Shifting k bits has the effect of multiplying 2^k as shown in Algorithm 1.

In this algorithm, "*HomCONSTANT*" is a function that produces one bit ciphertext corresponding to the input value

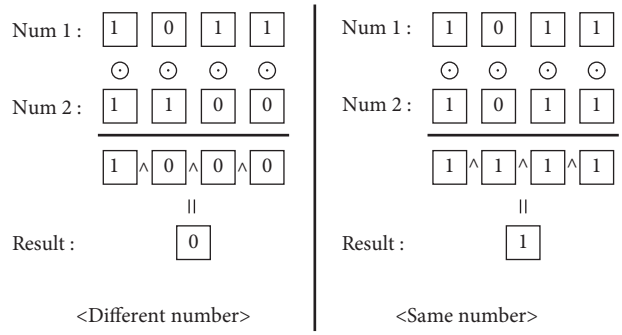


Figure 4: Example of equivalent comparison.

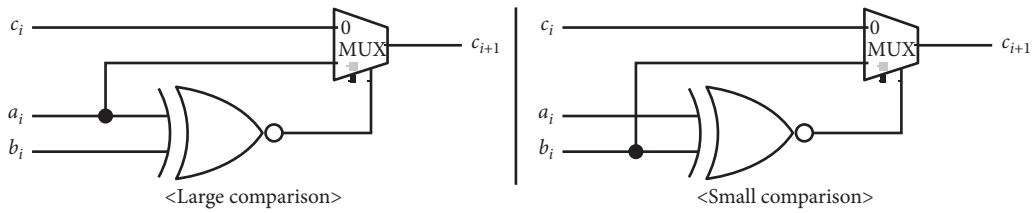


Figure 5: The circuit design for comparison.

Input: $a = [a_{l-1}, a_{l-2}, \ldots, a_0]$, k
Output: LSHIFT(a, k)
(1) **for** $i = 0 : (k-1)$ **do**
(2) $\quad a_i = HomCONSTANT(0)$
(3) **end for**
(4) **for** $i = k : (n-1)$ **do**
(5) $\quad a_i = HomCOPY(a_{i-k})$
(6) **end for**
(7) **return** $[a_{l-1-k}, \ldots, a_0, \mathbb{E}[0], \ldots, \mathbb{E}[0]]$

Algorithm 1: Pseudocode of left shift.

and "*HomCOPY*" is a function that produces the same one bit ciphertext as the result of the decryption, but different ciphertexts.

The right shift can be divided into a general shift, which is a method of shifting the upper k bits to $\mathbb{E}[0]$ after shifting like a left shift, and an arithmetic shift which shifts the upper k bits to the same value as the sign bit. An arithmetic shift is mainly used, and shifting k bits has the effect of dividing by 2^k.

4.6. Absolute Value Operation. In the plaintext, the absolute value algorithm outputs as it is if the most significant bit is 0 and takes the complement of 2 if the most significant bit is 1. Since the value of the most significant bit is not known in a ciphertext, a new algorithm must be designed. Let the original value be a and the value obtained by taking the complement of 2 to a be b; then, one is positive and the other is negative (except for 0). Now, let sign bit of a be a multiplexer factor, which returns a or b depending on the value:

$$|a| = \mathrm{HomMUX}(\mathrm{msb}(a), a, b). \tag{2}$$

4.7. Multiplication Operation. In general multiplication, multiplying m bits by n bits results in $(m+n)$ bits. When the two numbers to be multiplied are positive, the multiplicand is multiplied from the lsb of the multiplier to the upper bit as if it were calculated by hand. Then, the result of multiplication is the sum of all the left shifted values as the bit position of the multiplier increases. Thus, the smaller 1-bit of the multiplicand is, the more efficient it is. Therefore, we divide the multiplier by addition or subtraction to reduce the number of 1-bit as much as possible. However, as mentioned earlier, this is an algorithm that can be applied only to positive numbers, so a more advanced form of algorithm is needed to consider negative numbers. This is because, in the case of the unencrypted plaintext data, the sign of the data can be inspected by checking the msb, but in the case of the encrypted data, the value of the msb cannot be confirmed. That is, a new algorithm should be designed to output the correct result regardless of the sign of the given data. To solve this problem, we can calculate the product of positive numbers through an absolute value operation and then perform a 2's complement operation on the result according to the sign. That is, for multiplication of a and b, we follow the below way:

$$\begin{aligned} \mathrm{msb}(a) \oplus \mathrm{msb}(b) &= p, \\ M &= |a| \times |b|, \\ M' &= \text{2's complement of } M, \\ a \times b &= \mathrm{HomMUX}(p, M', M). \end{aligned} \tag{3}$$

Therefore, our algorithm adopts the latter method, and its circuit diagram is shown in Figure 6.

4.8. Division Operation. Binary division algorithms can be thought of as dividing input into positive cases and negative

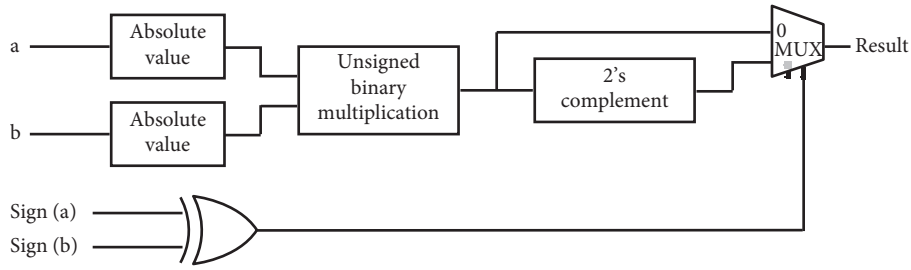


Figure 6: The circuit design for signed multiplication.

cases. First, let us consider the case where both the divisor and the dividend are positive. Let the array M, Q, and A have the same length of l, and initialize M to divisor, Q to dividend, and A to zero. The count value is the dividend length, l. And let $AQ = [A||Q]$ with a length of $2l$ and start the main part of the algorithm.

If the divisor or dividend is negative, we need to use a slightly different algorithm. First, we can implement negative binary division algorithm by modifying Algorithm 2 slightly. However, since the sign of the input value cannot be known, when the negative binary division algorithm is implemented with a new algorithm, both algorithms must be performed and a single result should be output according to the sign of the input value. This is inefficient because it takes time to perform Algorithm 2 twice. Therefore, we will implement the signed binary division algorithm using a second method that uses absolute values and multiplexer function as in multiplication. That is, for signed binary division M and Q, we follow the following way:

$$\begin{aligned} \mathrm{msb}(M) \oplus \mathrm{msb}(Q) &= p, \\ D &= \text{positive binary division}(|M|, |Q|), \\ D' &= \text{2's complement of } D, \\ \frac{Q}{M} &= \mathrm{HomMUX}(p, D', D). \end{aligned} \tag{4}$$

5. Experiments

5.1. Basic Gate Experiment. We implemented the operations of Section 4 based on the basic gates and checked the speed of 1-bit basic gate operation in TFHE 1000 times.

As shown in Table 2, the basic gates except the NOT gate have the same speed, and the speed of the NOT gate is significantly lower than that of the other gates. Also, the multiplexer function is implemented differently from the basic gates so that there is a difference in speed. It can be seen that the speed of the multiplexer function is faster than the speed of computing basic gate about two times.

5.2. Number of Gates Used in Designed Homomorphic Atomic Operations. Since all gates except NOT gate and MUX gate have the same speed, we will denote execution time of these gates as T_{G}. Time of the MUX gate is represented by T_{M}, and the NOT gate is omitted because the speed converges to zero. Table 3 shows the number of gates used when performing designed homomorphic operations with l-bit input values for each operation.

Most of the operations listed in Table 3 are linear for data length. In shift operation, the position of bit is shifted without using a gate operation, and the number of gates in multiplication and division operations is proportional to the square of the data length.

5.3. Execution Time of the Homomorphic Atomic Operations. In Table 4, we measure the speed of the operations based on 16 bits. The speed of the shift operation is not measured because gate is not used; for nonlinear operations, we measured 8, 16, and 32 bits to see the change in speed.

Looking at the measured values, the doubling of the length of the data increases the speed of both algorithms by about four times. This is because the speed of addition, subtraction, and comparison operations constituting the multiplication and division is linearly increased with respect to the data length, and the number of iterations of the algorithm is also proportional to the length of the data.

6. Applications

6.1. Linear Regression. Given a d-dimensional input variable $\mathbf{x}^{(i)} \in \mathbb{R}^d$ and its corresponding target variable $\mathbf{y}^{(i)} \in \mathbb{R}$ for $i = 1, 2, \ldots, n$, an inference on parameters of the linear function within hypotheses is defined as

$$h_\theta\left(\mathbf{x}^{(i)}\right) = \theta^{\mathrm{T}}\mathbf{x}^{(i)} = \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \cdots + \theta_d x_d^{(i)}, \tag{5}$$

for $\mathbf{x}^{(i)} = [1, x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)}]^{\mathrm{T}}$, parameters $\theta = [\theta_0, \theta_1, \theta_2, \ldots, \theta_d]^{\mathrm{T}}$, and number of features, $d + 1$. This regression describes a hyperplane in the d-dimensional space of the independent variables $\mathbf{x}$.

In general, the linear regression can be easily estimated by using least square estimation as follows:

$$\widehat{\theta} = \left(\mathbf{X}\mathbf{X}^{\mathrm{T}}\right)^{-1}\mathbf{X}\mathbf{Y}^{\mathrm{T}}, \tag{6}$$

where $\mathbf{Y} = [\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(n)}]$, $\mathbf{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}]$. However, in FHE, it is rather difficult to design and implement the inversion matrix of equation (6). Therefore, instead of the exact solution, we choose an approximation estimation which is based on the gradient descent update in order to avoid the calculation of inverse matrix.

The approximation estimation uses error function to optimize the parameters of both simple and multiple linear regression as follows:

Input: divisor M, dividend Q
(1) Shift the AQ to the left by one bit and let the upper l bit of AQ at A.
(2) Calculate $A - M$ and put it in A.
(3) If A is negative, the last bit of AQ becomes 0 and $A + M$ is calculated and put it in A to return to the value before step 2.
(4) If A is positive or zero, the last bit of AQ is 0.
(5) The count value is decremented by 1.
(6) If the count is not 0, the algorithm goes to step 1 and the algorithm is progressed.
(7) If the count value is 0, the result of algorithm is output (the lower l bit of AQ becomes the quotient and the upper l bit becomes the remainder).

Algorithm 2: Positive binary division operation.

Table 2: Execution time of basic gates (s).

AND	OR	NAND	NOR	XOR	XNOR	NOT	MUX
11.9	11.9	11.9	11.9	11.9	11.9	0.000162	22.4

Table 3: Time complexity of designed homomorphic atomic operation with l-bit input values.

Operation	Time complexity of designed operations
Addition	$(5l-3)T_G$
2's complement	$(2l-3)T_G$
Subtraction	$(5l-3)T_G$
Equivalent comparison	$(2l-1)T_G$
Large (small) comparison	$lT_G + lT_M$
Shift	≈ 0
Absolute value	$2lT_G + lT_M$
Multiplication	$(6l^2+4)T_G + 4lT_M$
Division	$(8l^2-4l+4)T_G + (l^2+2l)T_M$

Table 4: Execution time of designed homomorphic atomic operation.

Operation	Estimation (s)	Execution (s)	Error (%)
Addition	0.916	0.917	0.10
2's complement	0.345	0.351	1.73
Subtraction	0.916	0.919	0.32
Equivalent comparison	0.369	0.375	1.62
Large (small) comparison	0.548	0.548	0
Absolute value	0.703	0.712	1.28
Multiplication_8	5.334	5.396	1.11
Multiplication_16	19.759	19.898	0.70
Multiplication_32	76.028	77.413	1.82
Division_8	7.551	7.772	2.92
Division_16	30.108	30.553	1.47
Division_32	120.38	121.781	1.16

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}\left(h_\theta\left(\mathbf{x}^{(i)}\right) - \mathbf{y}^{(i)}\right)^2. \tag{7}$$

The main goal of linear regression is to fit a straight line through the data, so we minimize the error function $J(\theta)$. Gradient descent is achieved by an algorithm that starts with an initial θ and repeatedly performs the update:

$$\theta_j := \theta_j - \frac{\alpha}{n}\sum_{i=1}^{n}\frac{\partial}{\partial\theta_j}J(\theta), \tag{8}$$

where α is denoted by a learning rate. The parameters θ_j are updated concurrently for every iterations till convergence. Our algorithm of linear regression is given in Algorithm 3.

The method of implementing the linear regression is very similar to operation in the plaintext. However, it is calculated in an encrypted state; therefore, in an encrypted domain, we can calculate all operations in gradient descent algorithm which includes multiplication, addition, and subtraction operations. We initialized parameters θ to 0 and updated our parameters using linear regression function with FHE operations.

Input: data $\mathbf{X} \in \mathbb{R}^{D\times N}$, $\mathbf{Y} \in \mathbb{R}^{N}$, learning rate α, number of iteration
Output: Parameter $\theta \in \mathbb{R}^{D}$
(1) Initialize parameter θ to FX
(2) Gradient descent part 1: calculate partial derivative of cost function $J(\theta)$
(3) Gradient descent part 2: multiply α with the value of part 1
(4) Gradient descent part 3: update θ until iteration times
(5) return each of θ's

Algorithm 3: The algorithm of linear regression.

6.1.1. Performance Evaluation of FHE Linear Regression. We performed two experiments with varying d, the simple linear regression ($d = 1$) and the multiple linear regression ($d > 1$). We set the number of data (N), the number of dimensions (d), the length of data (l), and the number of iterations of the algorithm (p) as factors for the linear regression algorithm. Then, the number of gates (T) can be expressed as follows:

$$T(N,d,l,p) = 2Np\Big[\Big\{d\Big(6l^2+4\Big)T_{\mathrm{G}} + 4lT_{\mathrm{M}}\Big\} + (d+1)(5l-3)T_{\mathrm{G}}\Big] + 2(d+1)\Big\{\Big(6l^2+4\Big)T_{\mathrm{G}} + 4lT_{\mathrm{M}}\Big\} + (d+1)(5l-3)T_{\mathrm{G}}. \tag{9}$$

For the simple linear regression, we set the initial values to (N, d, l, p) = (10, 1, 16, 1) for the experiment. The dataset consists of a feature vector $x = [2, 4, 5, 6, 8, 10, 13, 16, 17, 19]$ and a target variable $y = [5, 9, 12, 14, 15, 18, 24, 26, 30, 32]$ with 10 data created artificially, and it takes 554 seconds with 0.01 running rate. The iteration proceeded 100 steps to converge $\theta = (3.404, 1.484)$ with threshold value, $\varepsilon = 0.1$.

For the multiple linear regression ($d > 1$), we set the initial values to (N, d, l, p) = (10, 2, 16, 1) for the experiment. The dataset consists of feature vectors $x_1 = [2, 4, 5, 6, 8, 10, 13, 16, 17, 19]$, $x_2 = [3, 5, 6, 7, 8, 11, 14, 15, 18, 20]$, and a target variable $y = [5, 9, 12, 14, 15, 18, 24, 26, 30, 32]$ with 10 data created artificially, and it takes 1047 seconds with 0.01 running rate. The iteration proceeded 50 steps to converge $\theta = (-0.952, 1.094, 3.331)$ with threshold value, $\varepsilon = 0.1$.

6.2. Logistic Regression. Implementation of various algorithms such as linear regression can be easily facilitated by our FHE arithmetic operations. However, logistic regression is an algorithm that holds a nonlinear function which requires variation in the equation to be calculated. Therefore, the key point of deriving FHE logistic regression lies in designing a nonlinear sigmoid function. We initially elaborate a brief derivation and structure of FHE logistic regression followed by explaining two ways of constructing logistic function.

Given an input variable $\mathbf{x}^{(i)} \in \mathbb{R}^{d}$ and its corresponding target variable $\mathbf{y}^{(i)} \in \mathbb{Z}_2$ for $i = 1, 2, \ldots, n$, an inference on parameters of the logistic function $g(z)$ within hypotheses is defined as

$$g(z) = \frac{1}{1+e^{-z}}, \quad z = \theta^{\mathrm{T}}x^{(i)}, \tag{10}$$

where $\theta^{\mathrm{T}}\mathbf{x}^{(i)} = \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \cdots + \theta_d x_d^{(i)}$ for $\mathbf{x}^{(i)} = [1, x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)}]^{\mathrm{T}}$, $\theta = [\theta_0, \theta_1, \theta_2, \ldots, \theta_d]^{\mathrm{T}}$, and number of features, $d + 1$. We also denote $x_j^{(i)}$ as an element of a matrix in the i-th row and j-th column position.

The logistic regression uses likelihood function to make an estimate on weight θ. If we let $p(y^{(i)} = 1 \mid \mathbf{x}^{(i)}; \theta) = h_\theta(\mathbf{x}^{(i)})$ and $p(y^{(i)} = 0 \mid \mathbf{x}^{(i)}; \theta) = 1 - h_\theta(\mathbf{x}^{(i)})$, the likelihood for a single data $\mathbf{x}^{(i)}$ is. $p(y^{(i)} \mid \mathbf{x}^{(i)}; \theta) = (h_\theta(\mathbf{x}^{(i)}))^{y^{(i)}}(1 - h_\theta(\mathbf{x}^{(i)}))^{1-y^{(i)}}$.

Finally, the likelihood function for the whole data, $\{x^{(i)}\}_{i=1}^{n}$, is to multiply likelihood of each data. Next, log operation is performed to enumerate log likelihoods in a linear combination as the follows:

$$L(\theta) = \sum_{i=1}^{n} y^{(i)}\log h_\theta\Big(x^{(i)}\Big) + \Big(1 - y^{(i)}\Big)\log\Big(1 - h_\theta\Big(x^{(i)}\Big)\Big). \tag{11}$$

In order to maximize the likelihood, $L(\theta)$, we chose to perform gradient descent algorithm that iteratively updates cost function, $J(\theta)$, where $J(\theta) = -L(\theta)$. Therefore, θ is updated with the following equation:

$$\theta_j := \theta_j - \frac{\alpha}{n}\sum_{i=1}^{n}\Big(h_\theta\Big(x^{(i)}\Big) - y^{(i)}\Big)x_j^{(i)}. \tag{12}$$

Existing literature [18] designed a nonlinear logistic function by two approximation techniques, namely, the Taylor series method and least square approximation. In this paper, we show feasibility of constructing two different approximation techniques based on our proposed bitwise FHE operations to perform the logistic regression.

6.2.1. Taylor Series Method. It is well-known that Taylor expansion enables a differentiable real-valued function $f(x)$ to be expanded in a series at $x = a$ such that $f(x) = \sum_{r=1}^{\infty} f^{(r)}(a)/r!\,(x-a)^r = f(a) + (f'(a)/1!)(x-a) + (f''(a)/2!)(x-a)^2 + \cdots$ where $f^{(r)}$ is denoted by r-th derivative of f.

Bos et al. applied Taylor series expansion to logistic function which facilitates calculation of the nonlinear function since the altered equation incorporates only the four fundamental operations [18]. Therefore, Taylor series polynomial of degree 9 for sigmoid function can be derived as

$$g(x) = \frac{1}{1+e^{-x}} \approx \frac{1}{2} + \frac{1}{4}x - \frac{1}{48}x^3 + \frac{1}{480}x^5 - \frac{17}{80640}x^7 + \frac{31}{1451520}x^9. \tag{13}$$

Using our basic bitwise FHE operations that are presented in the previous section, we can construct approximate logistic function by Algorithm 4. In addition, we refer c_i to coefficients of $g(x)$ where $c_0 = 1/2$, $c_1 = 1/4 \cdot, \cdot$, $c_5 = 31/1451520$.

Figure 7 illustrates approximated logistic function with respect to Taylor series expansion. Our approach guarantees a boundary of $(-1, 1)$ while $(-2, 2)$ for the existing literature [18]. This is due to the 4th and 5th coefficients that are 0 for the length of the input designated by 32 bit. This can be solved by assigning larger length to represent the coefficient numbers.

6.2.2. Least Square Approximation. Kim and Cheon et al. proposed a least square polynomial that broadens bounded domain of Taylor series expansion to $(-8, 8)$ [19, 20]. The underlying principle is to derive a function $g(x)$ that minimizes mean squared error (MSE) such that $1/|I| \int_I (g(x) - f(x))^2 dx$ where $|I|$ is denoted by the length of an interval.

We omit an algorithm for implementing the least square approximation with respect to our scheme since the algorithm follows a similar procedure as in Algorithm 4. The visualized comparison of the real sigmoid function with our approach and that of the existing literature [18] can be seen in Figure 8 to verify that our FHE scheme can approximate the desired function equal to the current literature.

6.2.3. FHE Gradient Descent Algorithm. When the logistic function is designed either by the Taylor series or the least square approximation technique, we are able to perform the gradient descent algorithm for parameter estimate. The process of logistic regression is indicated in Algorithm 5.

6.2.4. Performance Evaluation of the FHE Logistic Regression. We implemented logistic regression with two of the strategies mentioned previously. From Algorithm 4, we claim that number of data (N), length of data (l), dimension (d), and iteration (p) are the principal factors of time complexity (T) for both methods. We deliver their time performances in a precise manner, where T_{Taylor} and T_{ls} are time complexity of the Taylor series and least square approximation, respectively:

$$\begin{aligned} T_{\text{taylor}}(N, l, d, p) &= Ndp\left[13\left\{\left(6l^2 + 4\right)T_{\text{G}} + 4lT_{\text{M}}\right\} + 6(5l-3)T_{\text{G}}\right] + 6(5l-3)T_{\text{G}}dp, \\ T_{\text{ls}}(N, l, d, p) &= Ndp\left[10\left\{\left(6l^2 + 4\right)T_{\text{G}} + 4lT_{\text{M}}\right\} + 5(5l-3)T_{\text{G}}\right] + 6(5l-3)T_{\text{G}}dp. \end{aligned} \tag{14}$$

Since the time for experiment requires fairly significant amount of time, we set number of data, dimension, and iteration to be 10, 2, and 1, respectively. The summary of time performance with respect to 16 bit is elaborated in Table 5.

6.3. kNN Classifier. The bitwise FHE method of implementing the kNN algorithm in Algorithm 6 is almost similar to that of the plaintext, except the sorting operation which is described in the next section. The conventional kNN algorithm uses Euclidean distance between data, but our algorithm replaced the distance as the sum of the absolute value for speed efficiency. Also, when sorting the calculated distances, we searched for only the k smallest values to reduce the computation time. As shown in Algorithm 6, we need to design two additional homomorphic operations for the homomorphic kNN classifier: sort of Algorithm 7 and conditional swap of Algorithm 8.

When sorting is completed, we check the labels of the nearest k data and output the major labels. Since the label is also encrypted, it is not possible to know which label is the most major.In order to attain the most frequently used label, we first counted number of data with the same label. Since the counting numbers are encrypted, we perform equivalent compare operation of a label to the other labels. Lastly, we add all the output numbers and sort out in descending order to pick the largest number, which is our desired label. Algorithm 9 represents the pseudocode that finds the most major label among the labels of k-nearest data in our kNN algorithm.

6.3.1. Sorting for kNN Algorithm. The kNN algorithm on encrypted domain requires sorting algorithm to find the nearest neighbors, so we design a new sort algorithm for ciphertext. Algorithm 7 represents the pseudocode to sort the numbers in arr[n] by the selection sort algorithm.

A swap operation that simply exchanges a location in a ciphertext should only change its position as in plaintext, but to apply the selection sort algorithm to ciphertext, we must decide whether to relocate it through a large or small comparison. So, we have to input the factor to determine whether to swap or not, and we call this swap operation conditional swap.

6.3.2. Conditional Swap. Conditional swap operation runs swapping if a determining factor is $E[1]$. Otherwise, data are not swapped. In the selection sort algorithm, if arr[i] is bigger than arr[j], it has to be swapped. Therefore, it outputs $E[1]$ through a large comparison operation and puts it into the factor to decide whether to swap or not. If arr[i] is less than or equal to arr[j], swap will not occur because it outputs $E[0]$ through the large comparison operation. Algorithm 8 represents the conditional swap pseudocode that takes this situation into consideration.

When S is $E[1]$, arr[i] = NS arr[i] + S arr[j] is arr[j] and arr[j] = S arr[i] + NS arr[j] is arr[i]. Thus, swap operation has occurred. The other way, in case $S = E[0]$, arr[i] = NS arr[i] + S arr[j] is arr[i] and arr[j] = S arr[i] + NS arr[j] is arr[j]. Thus, swap operation has not occurred.

Input: a training data $\mathbf{x}^{(i)}$
Output: logistic value of $\mathbf{x}^{(i)}$ w.r.t the Taylor expansion method
(1) Convert coefficient c_i into arrays
(2) Construct power series of x to 9th power
(3) Multiply c_i with corresponding power of x
(4) Add all the derived terms in step 3

Algorithm 4: FHE sigmoid function by Taylor expansion.

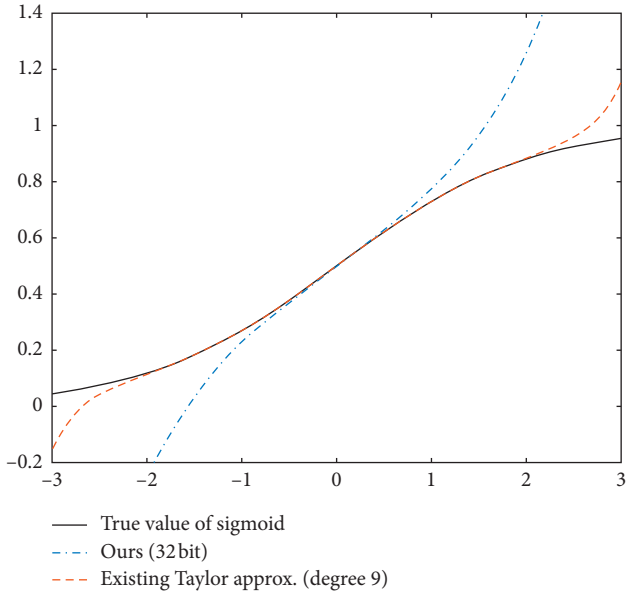


Figure 7: Comparison of real sigmoid with our approach and Taylor series approximation from existing literature [18].

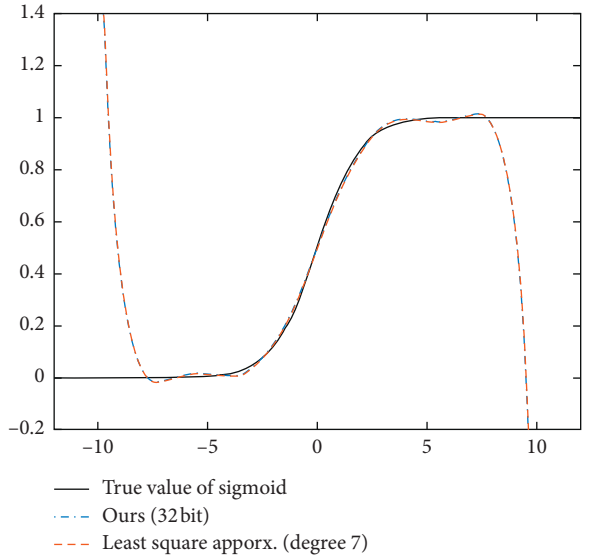


Figure 8: Comparison of real sigmoid with our approach and least square approximation.

6.3.3. Performance Evaluation of the FHE kNN Classifier. We set the number of data (N), the dimension of data (d), the length of data (l), the number of near neighbors (k), and the length of label (L) as factors of the kNN algorithm. Then, the time complexity of kNN algorithm (T) can be expressed as follows:

$$\begin{aligned} T(N,d,l,k,L) = N\big[\{d(12l-6)-5l+3\}T_{\mathrm{G}} \\ + dT_{\mathrm{M}}\big] + \frac{k^2+k(2N-3)}{2}\{lT_{\mathrm{G}} \\ + (3l+2L)T_{\mathrm{M}}\} \\ + \frac{(k-1)(k-2)}{2}(7L-4)T_{\mathrm{G}} \\ + (k-1)\left(LT_{\mathrm{G}}+5LT_{\mathrm{M}}\right). \end{aligned} \tag{15}$$

We set the initial values to $(N, d, l, k, L) = (64, 1, 10, 3, 1)$ for the experiment. When conducting experiment with the initial value, it took 226 seconds. Then, we performed the experiment by changing the value of each factor one by one. As a result, because the algorithm consists solely of linear operations except k, we confirmed that the speed of the algorithm is almost proportional to the value of each factors.

6.4. k-Means Algorithm for Image Segmentation. We also performed gray color image segmentation using the k-means algorithm. The target image for the homomorphic segmentation has the 8-bit gray color of each pixel in the image, and the k-means algorithm is used to input the encrypted color value of all the pixels. In order to do this, the cloud server first obtains encrypted values of the pixels at N random locations rather than all encrypted pixels for efficient computation. Afterwards, the k-means algorithm is applied to partition N encrypted pixels into k clusters. As a result, the cloud server calculates the representative values of k clusters in a homomorphic way. After deciphering the representative values in the client's side, the colors of all the pixels in the image are compared with the representative values, and image segmentation is performed by replacing the color with the representative value of the near cluster. Our algorithm of k-means is given in Algorithm 10; we performed the algorithm by expanding the total data size to 10 bits considering 8-bit original data, the sign bit, and addition operation.

In general, use the Euclidean distance when calculating the distance between two points. In this experiment, however, another method can be used because the dimension of

Input: training data $\mathbf{X}, \mathbf{Y}$
Output: parameter θ
(1) Set parameter θ to $\mathbf{0}$
(2) Assign learning rate α and iteration number p, respectively
(3) Calculate partial derivative of cost function $J(\theta)$ for the training data $\mathbf{X}, \mathbf{Y}$
(4) Multiply α/n by the previous outcome
(5) Update θ by the result of step 4
(6) Repeat steps 3 to 5 for p times to obtain θ

Algorithm 5: The algorithm of logistic regression.

Table 5: Execution time of logistic regression w.r.t two different strategies (16-bit inputs).

Strategy	Taylor expansion series	Least square approximation
Time (s)	12961	11315

Input: training data ($\mathbf{X}$, $\mathbf{Y}$, $\mathbf{l}$), test data ($\mathbf{x}$, $\mathbf{y}$), and the number of neighbors, k
Output: test label l_t
(1) Calculate distance with training data ($\mathbf{X}$, $\mathbf{Y}$) and test data ($\mathbf{x}$, $\mathbf{y}$) with absolute value operation.
(2) Sort the smallest k distance using conditional swap operation on selection sort algorithm.
(3) Output most major label among the labels of nearest k data.

Algorithm 6: The algorithm of kNN classification.

Input: $\text{arr}[n] = [a_1, a_2, \ldots, a_n]$
Output: SORT($\text{arr}[n]$)
(1) **for** $i = 1 : (n-1)$ **do**
(2) **for** $j = (i+1) : (n-1)$ **do**
(3) COND_SWAP($\text{arr}[i], \text{arr}[j], S$)
(4) **end for**
(5) **end for**
(6) **return** $\text{arr}[n]$

Algorithm 7: Pseudocode of sorting.

Input: $\text{arr}[i]$, $\text{arr}[j]$, S
Output: COND_SWAP($\text{arr}[i], \text{arr}[j], S$)
(1) S = L_COMP($\text{arr}[i], \text{arr}[j]$): large comparison
(2) $NS = S$
(3) $\text{arr}[i] = NS \wedge \text{arr}[i] + S \wedge \text{arr}[j]$
(4) $\text{arr}[j] = S \wedge \text{arr}[i] + NS \wedge \text{arr}[j]$
(5) **return** $\text{arr}[i], \text{arr}[j]$

Algorithm 8: Pseudocode of conditional swap.

the data is one-dimensional. After calculating the center point of each representative value of each cluster, labeling is performed without calculating the distance through comparison of the data with the values.

Since the values of the data are not known in the encrypted state, when the representative values of the cluster are given, it is not known which value is closest to the representative value. However, we can set the label to distinguish the nearest value from the representative value of each cluster. Let $E[1]$ and $E[0]$ denote each label. Through an AND operation of each data and its label, we can divide all data into $\mathbf{0}$ of which all bits are set to $E[0]$ and non-$\mathbf{0}$ values (if a data's label is $E[0]$, the result of AND operation is $\mathbf{0}$ value; otherwise, the result is non-$\mathbf{0}$ value). Now, we can

Input: $l_1, l_2, \cdots, l_k$
Output: l_p
(1) **for** $i = 1 : (k-1)$ **do**
(2) $s_i = 1$
(3) **for** $j = (i+1) : k$ **do**
(4) **if** $l_i = l_j$ **then**
(5) $c = 1$
(6) **else**
(7) $c = 0$
(8) **end if**
(9) $s_i = s_i + c$
(10) **end for**
(11) **end for**
(12) $p = \arg_i \max(s_i)$ for $i = 1 : k$
(13) **return** l_p

Algorithm 9: The pseudocode to find a label with majority.

Input: data **X**, the number of neighbors k, and the initial value of clusters u_k
Output: labeled data (**X**, **l**)
(1) Obtain the distance between each cluster u_k and the data **X**.
(2) For each data, label closest clusters.
(3) Initialize the cluster u_k by averaging the data with the same cluster value.
(4) Repeat steps 1 to 3 to obtain converged clusters and return the labeled data.

Algorithm 10: The algorithm of k-means.

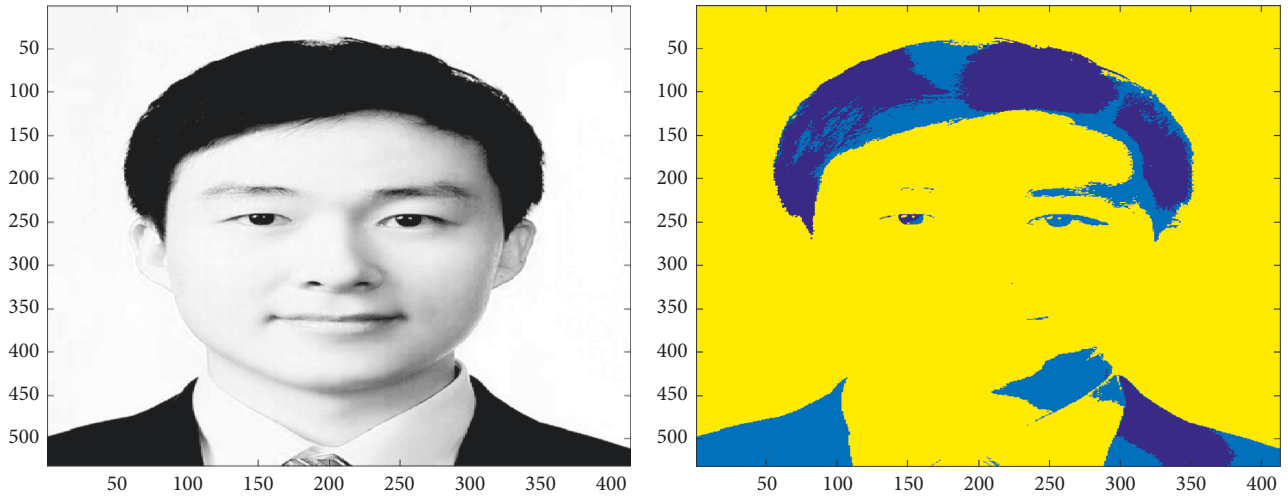


Figure 9: (a) An original image and (b) its segmented image on the encrypted domain.

obtain the average of the clusters by dividing sum of data with sum of labels.

6.4.1. Performance Evaluation of the FHE k-Means Clustering Algorithm. We set the number of data (N), the length of data (l), the number of clusters (k), and the number of iterations of the algorithm (p) as factors for the k-means algorithm. Then, the number of gates (T) can be expressed as follows:

$$\begin{aligned} T(N,l,k,p) = p\big[&(k-1)(5l-3)T_{\mathrm{G}} \\ &+ N(k-1)\left(T_{\mathrm{G}} + T_{\mathrm{M}}\right) \\ &+ k\{NlT_{\mathrm{G}} + (N-1)(5l + 5\log N - 6)T_{\mathrm{G}}\} \\ &+ k\left\{\left(8l^2 - 4l + 4\right)T_{\mathrm{G}} + \left(l^2 + 2l\right)T_{\mathrm{M}}\right\} \\ &+ NlkT_{\mathrm{G}}\big]. \end{aligned} \tag{16}$$

The algorithm took 148 seconds given the initial value, $(N, l, k, p) = (64, 10, 3, 1)$. The experiment was set up with an

input of an image in Figure 9(a), where the parameters are given as 64, 10, 3, and 10. The representative value for each cluster is recorded as 23, 56, and 170, respectively. The experiment took approximately 1,500 seconds, and the result of segmentation can be checked in Figure 9(b).

7. Discussion

In this section, we describe the limitation of our proposed approach in usage. Our proposed approach has a concern: it has extremely slow computation with large memory space.

Currently, it is true that bit-based schemes are inefficient in terms of speed and memory compared to integer-based schemes. However, integer-based schemes have a fatal disadvantage that their possible operations are limited and can only be used for specific algorithms. This is a fundamental problem and hard to improve. On the other hand, the speed of computation, which is a disadvantage of bit-based schemes, can be improved more flexibly.

Our current approach is not optimized yet, so each operation on encrypted domain is extremely time consuming. However, this problem may be addressed by accelerating the computation with a lot of state-of-the-art techniques. For instance, the atomic operations can be implemented in a hardware level rather than in a software level. FPGA and ASIC would be the good candidates for the implementation. Additionally, we can reduce the computation time by optimizing the logic and programming codes in a software level. We can also save the computation time using a graphical processing unit (GPU) and parallel computing scheme.

8. Conclusion

In this paper, we have proposed basic homomorphic arithmetic operations using bitwise homomorphic gates. We applied these bitwise homomorphic operations to several well-known data mining techniques: the linear regression, logistic regression, k-NN classifier, and k-means clustering. To implement the algorithms, we introduced advanced bitwise operations such as sorting and conditional swap, which are specific to bitwise homomorphic operations. With our proposed bitwise homomorphic atomic and additional operations, even data scientists without any knowledge of FHE can easily analyze and process data on encrypted domain.

Data Availability

The training data and image data used to support the findings of this study have not been made available because it is artificially created and also small enough so that the reader can easily create it.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study was supported by the Institute for Information and Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (no. 2017-0-00545).

References

[1] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 24–43, Springer, Berlin, Germany, 2010.

[2] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 831–871, 2014.

[3] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the STOC*, vol. 9, pp. 169–178, Washington, DC, USA, May 2009.

[4] D. Li, C. Liu, and W. Gan, "A new cognitive model: cloud model," *International Journal of Intelligent Systems*, vol. 24, no. 3, pp. 357–375, 2009.

[5] T. Veugen, "Encrypted integer division," in *Proceedings of the IEEE International Workshop on Information Forensics and Security, WIFS 2010*, pp. 1–6, IEEE, Seattle, WA, USA, December 2010.

[6] T. Graepel, K. Lauter, and M. Naehrig, "Ml confidential: machine learning on encrypted data," in *Proceedings of the International Conference on Information Security And Cryptology*, pp. 1–21, Springer, Seoul, Korea, November 2012.

[7] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, pp. 334–348, IEEE, San Francisco, CA, USA, 2013.

[8] T. Veugen, "Encrypted integer division and secure comparison," *International Journal of Applied Cryptography*, vol. 3, no. 2, pp. 166–180, 2014.

[9] L. J. Aslett, P. M. Esperança, and C. C. Holmes, "Encrypted statistical machine learning: new privacy preserving methods," 2015, http://arxiv.org/abs/1508.06845.

[10] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *Proceedings of the NDSS*, San Diego, CA, USA, 2015.

[11] R. Gilad-Bachrach, N. Dowlin, K. Laine et al., "Applying neural networks to encrypted data with high throughput and accuracy," in *Proceedings of the International Conference on Machine Learning*, pp. 201–210, New York City, NY, USA, June 2016.

[12] R. L. Rivest, L. Adleman, M. L. Dertouzos et al., "On data banks and privacy homomorphisms," *Foundations of Secure Computation*, vol. 4, no. 11, pp. 169–180, 1978.

[13] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proceedings of the Advances in Cryptology—EUROCRYPT' 99*, vol. 99, pp. 223–238, Springer, Espoo, Finland, May 1999.

[14] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: Fast Fully homomorphic encryption library over the torus," *Journal of Cryptology*, pp. 1–58, 2017.

[15] L. Ducas and D. Micciancio, "FHEW: bootstrapping homomorphic encryption in less than a second," *Eurocrypt (1)*, vol. 9056, pp. 617–640, 2015.

[16] W. Lu, S. Kawasaki, and J. Sakuma, "Using fully homomorphic encryption for statistical analysis of categorical, ordinal and numerical data," in *Proceedings of the 2017 Network and Distributed System Security Symposium*, p. 1163, San Diego, CA, USA, 2017.

[17] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Faster packed homomorphic operations and efficient circuit bootstrapping for tfhe," in *Proceedings of the International Conference on the Theory And Application of Cryptology And Information Security*, pp. 377–408, Springer, Hanoi, Vietnam, December 2017.

[18] J. W. Bos, K. Lauter, and M. Naehrig, "Private predictive analysis on encrypted medical data," *Journal of Biomedical Informatics*, vol. 50, pp. 234–243, 2014.

[19] M. Kim, Y. Song, S. Wang, Y. Xia, and X. Jiang, "Secure logistic regression based on homomorphic encryption: design and evaluation," *JMIR Medical Informatics*, vol. 6, no. 2, 2018.

[20] J. H. Cheon, D. Kim, Y. Kim, and Y. Song, "Ensemble method for privacy-preserving logistic regression based on homomorphic encryption," *IEEE Access*, vol. 6, pp. 46938–46948, 2018.

Hindawi
Security and Communication Networks
Volume 2019, Article ID 6268476, 10 pages
https://doi.org/10.1155/2019/6268476

Research Article

Automated Dataset Generation System for Collaborative Research of Cyber Threat Analysis

Daegeon Kim and Huy Kang Kim

Graduate School of Information Security, Korea University, Seoul, Republic of Korea

Correspondence should be addressed to Huy Kang Kim; cenda@korea.ac.kr

Received 5 April 2019; Revised 30 July 2019; Accepted 5 September 2019; Published 30 September 2019

Guest Editor: Surya Nepal

Copyright © 2019 Daegeon Kim and Huy Kang Kim. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The objectives of cyberattacks are becoming sophisticated, and attackers are concealing their identity by masquerading as other attackers. Cyber threat intelligence (CTI) is gaining attention as a way to collect meaningful knowledge to better understand the intention of an attacker and eventually predict future attacks. A systemic threat analysis based on data acquired from actual cyber incidents is a useful approach to generating intelligence for such an objective. Developing an analysis technique requires a high-volume and fine-quality data. However, researchers can become discouraged by inaccessibility to data because organizations rarely release their data to the research community. Owing to a data inaccessibility issue, academic research tends to be biased toward techniques that develop steps of the CTI process other than analysis and production. In this paper, we propose an automated dataset generation system called CTIMiner. The system collects threat data from publicly available security reports and malware repositories. The data are stored in a structured format. We released the source codes and dataset to the public, including approximately 640,000 records from 612 security reports published from January 2008 to June 2019. In addition, we present a statistical feature of the dataset and techniques that can be developed using it. Moreover, we demonstrate an application example of the dataset that analyzes the correlation and characteristics of an incident. We believe our dataset will promote collaborative research on threat analysis for the generation of CTI.

1. Introduction

Cyber threat intelligence (CTI) is evidence-based knowledge including context, mechanisms, indicators, implications, and actionable advice regarding existing or emerging threats to assets [1]. CTI can be utilized to achieve a broad situational awareness, collaborate in defeating cyber threats faced by others, and prevent cyber threats by applying CTI into defense systems.

With an increase in global cyber threats, CTI is gaining increased attention as a response to such threats. Many nations and organizations have also attempted to promote the use of CTI by enacting laws that legalize and encourage the collection of CTI [2], sharing CTI through multilateral cooperation [3–5], and establishing various standards [6, 7]. Furthermore, during the recent decade, the number of articles related to CTI have dramatically increased, as shown in Figure 1 (Google Scholar search result with exact keyword matching of "cyber threat intelligence" including patents and citations on March 30, 2019).

During the Olympic Winter Games in PyeongChang 2018, a cyberattack targeting the server operated by the organizing committee occurred. What makes this case noteworthy is that the security researchers attributed different countries as the perpetrators of the attack. The authors in [4, 8] insisted that Chinese and Russian actors were responsible for the attack, respectively. In [9, 10], it was pointed out that it is impossible to attribute the attack to a specific country based on the small amount of code discovered, which overlaps malware used by the Lazarus Group, a hacking group from North Korea. However, the authors of [4] insisted that there was evidence indicating that a Russian attacker tried to masquerade as a North Korean hacking group. In this example, we can see that a precise

evidence-based analysis that considers all possibly related cases is vitally important for CTI generation.

However, among the traditional intelligence processes [11], i.e., planning and direction; collection, processing, and exploitation; analysis and production; and dissemination and integration, most technical studies on CTI have tended to focus on steps other than analysis and production, which require a real CTI dataset. Despite the many advantages of a CTI analysis as mentioned in [12], such as (1) an interoperability of the data (machine, vendor, and organization independent), (2) a compact expression of the heterogeneous source of the threat information, and (3) the possibility of conducting a long-term and nation-wide threat analysis, we believe that the most challenging aspect of such a study is the limited accessibility of data to researchers. Although some web services provide functionality when searching for threat data, they do not offer a sufficient and useful set of data for research purposes. In addition, most of the datasets consist of only specific data types, e.g., the IP, URL, or hash value, and some datasets are strictly restricted to access in certain regions or to people of a particular nationality.

In this paper, we propose a cyber threat dataset generation system called CTIMiner, which automatically collects data from public security reports and malware repository websites and stores the data in a structured format. The generated dataset contains several types of data including malware analysis information, which consists of the file path, mutex, code sign information, and the other data types listed above. The main contributions of our work are as follows:

(i) Promoting collaborative CTI analysis research by proposing a cyber threat data generation system and a public database

(ii) Demonstrating the use of the dataset for a correlation analysis

(iii) Suggesting the development of techniques to generate CTI from a dataset

At this point, it would be warranted to introduce the techniques used to generate CTI from a dataset. However, this is beyond the scope of this paper and remains as our future research concern. We believe that the suggestion of the required techniques for analyzing the dataset can inspire the researchers and promote research into CTI analysis.

The remainder of this paper is organized as follows. The intelligence process and its associations with CTI activities are presented in Section 2 with several studies related to each step. The overall system architecture of CTIMiner and the phases composing the runtime process are described in Section 3. The dataset structure, the data categories, and the statistical features are detailed in Section 4. In Section 5, the dataset usage is demonstrated, and analysis techniques are suggested. In Section 6, we provide some concluding remarks.

2. Related Works

2.1. Intelligence Process and Automated CTI Activities. In the field of military operation, the well-defined intelligence process illustrated in Figure 2 was adapted to efficiently generate intelligence from low-level data collected in the field to support the decision-making process. This process is intended to be followed by a human intelligence officer but can also be projected into automated CTI activities.

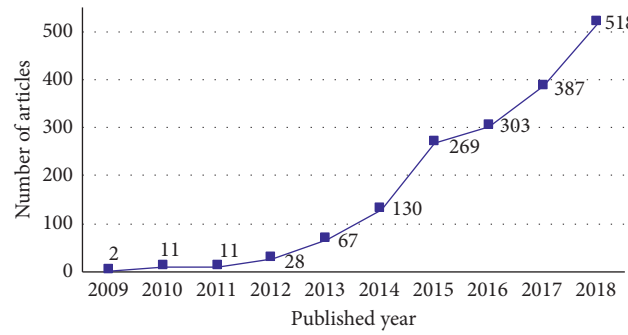


Figure 1: Number of articles related to CTI within the most recent decade.

Once the operation direction is determined to fulfill the identified intelligence requirement, the raw data are collected and extracted from the sensors and data sources, which have the ability and functionality to obtain such data. The data gathered from these various sources are combined and converted into forms, in other words information, allowing the data to be efficiently analyzed. The information is passed into an analysis algorithm, such as a big data or machine learning-based method, which enables the intelligence collected to be used by human analysts. Such intelligence is then spread to others who have access to it. The shared intelligence can also be integrated into the intelligence already available to users.

The association between the intelligence process and automated CTI activities is illustrated in Figure 3. In the following subsections, previous studies regarding CTI are introduced in order of the applied intelligence process, excluding the planning and direction steps because these are more strategic, rather than technical, concerns.

2.2. Collection. Because CTI is also a product of threat data processing through the intelligence process, low-level threat data can be collected during this step. Goel classified the types of data to be collected into unstructured and network data [13]. The former typically consists of hacker forum postings, blogs, and websites, whereas the latter is generated from information security systems such as firewalls, intrusion detection systems, and honeynets.

Benjamin et al. proposed a method for extracting information from hacker forums, IRC channels, and carding shops to identify threats [14]. In addition, Fachkha and Debbabi characterized the darknet and compared several methods for extracting threat information there [15].

As a data repository for research regarding cyber security analysis, the Information Marketplace for Policy and Analysis of Cyberrisk & Trust (IMPACT) [16], which is based on Protected Repository for the Defense of Infrastructure Against Cyber Threats (PREDICT) [17], provides several types of data, such as network flow, IDS and firewall, and

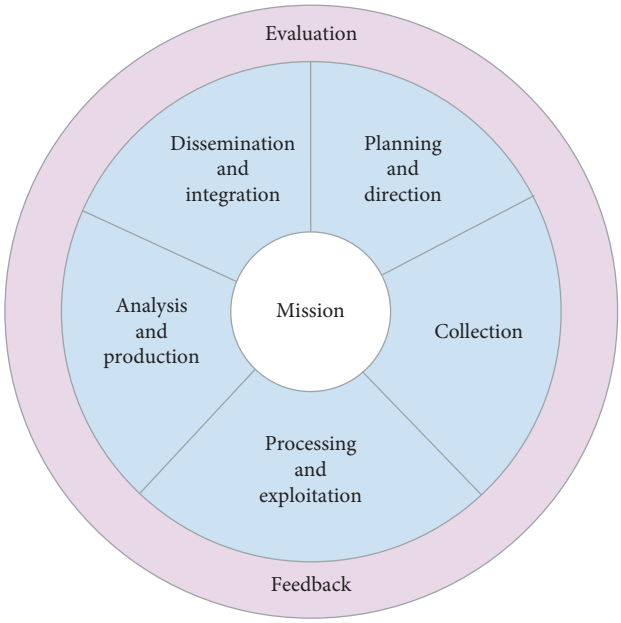


Figure 2: Intelligence cycle defined during military operation [11].

unsolicited email data. It also provides useful tools for data analysis. However, the service is only available to DHS-approved countries, namely, the United States, Australia, Canada, Israel, Japan, Netherlands, Singapore, and the United Kingdom.

2.3. Processing and Exploitation. During the processing and exploitation step, raw data collected are converted into forms that can be readily applied by intelligence analysts and other users. Unstructured data and heterogeneous sources of data having different structures can be stored in a unified data format during this step for further analysis.

STIX [18] and OpenIOC [19] proposed by MITRE and MANDIANT are representative standards for expressing threat data. Specifically, STIX is widely used owing to the scalability of its schema, which uses components such as CybOX and CAPEC. Liao et al. proposed an element extraction method for constructing structured data from unstructured data [20]. One notable aspect of this approach is that the meaning of the elements in the context can also be retrieved using a natural language processing technique.

2.4. Analysis and Production. During the analysis and production step, all processed information is integrated, evaluated, analyzed, and interpreted to produce intelligence. Kornmaier and Jaouën insisted that, to generate operational or strategic intelligence beyond tactical information, which is technical in nature, the threat data should be fused with data collected from different disciplines such as Human Based Intelligence (HUMINT), Imagery Intelligence (IMINT), Signal Intelligence (SIGINT), and Geographic Intelligence (GeoINT) [21].

Modi et al. proposed an automated threat data fusing system that correlates data crawled from the web by applying a string-matching based approach [22]. Similar commercial CTI services have also been developed, such as iDefense® IntelGraph by Verisign and a web intelligence engine by Recorded Future that allows users to navigate through extensive threat data following a string-matching correlation. One key feature of Recorded Future is that it can conduct a predictive analysis of specific future events through the use of information compiled in advance [23]. However, commercial services provide an indicator-centric analysis approach making it difficult to trace the correlation between incidents.

Kim et al. proposed a general framework for an efficient CTI correlation analysis by adopting a novel concept that expresses similarity between threat events in a graphical structure [12]. A graphical structure allows the analysts to trace the specifications and transition of related cyber incidents to infer an attacker's intention.

Using a threat report as the source of information, Qamar et al. proposed an automated mechanism to analyze the risk of a reported threat toward a networked system [24]. For this purpose, they defined the ontology of the IoCs, network, associated risk, and their relations. For the risk analysis of a networked system, four parameters, namely, the threat relevance, threat likelihood, total loss of affected assets, and threat reachability, are defined.

2.5. Dissemination and Integration. During the dissemination and integration step, intelligence is delivered to and used by the consumer. A guideline [25] and technical standard protocol [26] have been developed for sharing CTI. In addition, MISP [27], MANTIS [28], and CIF [29] are useful open-source platforms to store and share CTI. The closed CTI sharing ecosystems (e.g., C-TAS [30]) also exist to several countries and industry fields.

As more participants in a community share CTI, access control issues with the shared data often arise. Zhao and White proposed an access control model that extends the group-centric Secure Information Sharing model to support collaborative information sharing in a community [31]. Although such assistive technologies promote CTI sharing, for example, social and political issues, the authority to operate CTI sharing policies and the trust management within a community are often controversial when establishing collaborative CTI sharing.

2.6. Data, Information, and Intelligence. In many CTI-related studies, the terms, *data*, *information*, and *intelligence*, are often intermixed without clarification. We need to use them clearly, as illustrated in Figure 4, based on the definition in [11].

Data are the individual facts collected from sensors in an operational environment. *Information* is data gathered and processed into an intelligible form, and *intelligence* is the new understanding of current and past information that allows a prediction of the future and informed decisions.

These definitions are applied not only to the general intelligence process but also to CTI activities. Throughout the data fusion and mining process, Bass defined data as measurements and observations; information as data placed

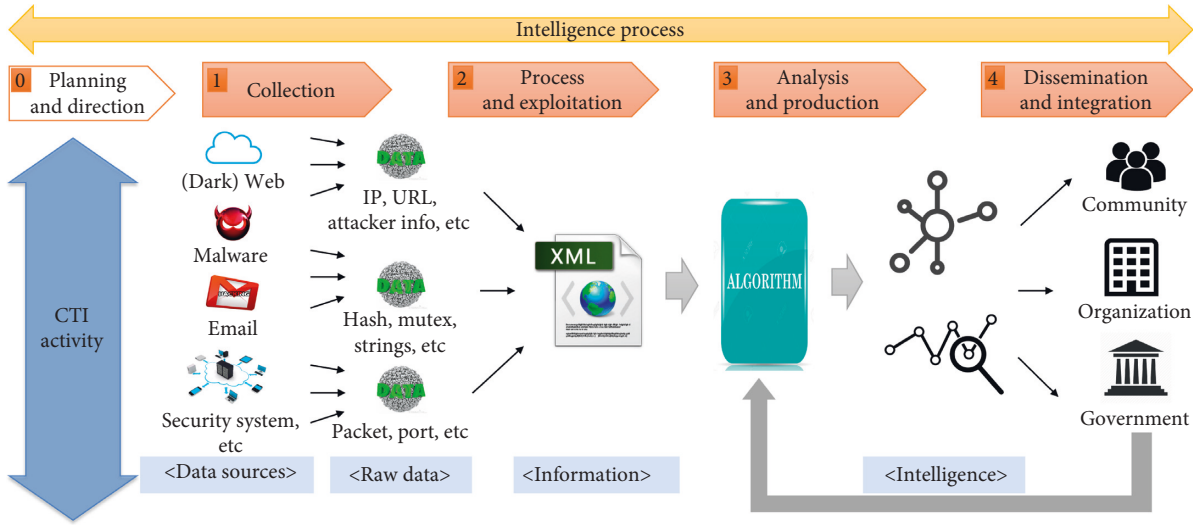


Figure 3: Association between the intelligence process and automated CTI activities.

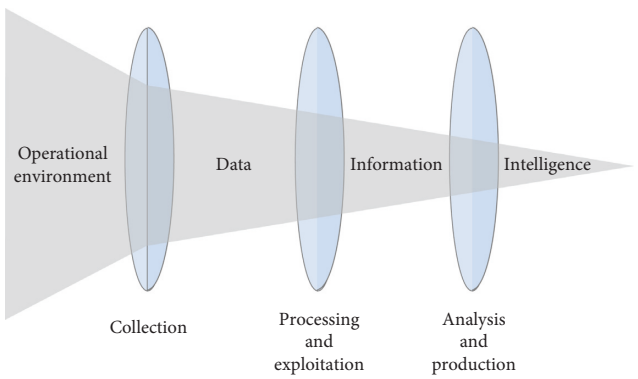


Figure 4: Data, information, and intelligence.

in context, indexed, and organized; and knowledge, which is equal to intelligence, as information that has been explained or understood [31].

3. CTIMiner System Architecture

We propose a cyber threat data collecting system, CTIMiner, using the system architecture presented in Figure 5. The CTI collecting procedure is composed of three phases. During the first phase, the system gathers threat data from publicly accessible cyber intelligence reports published by organizations and companies. It also collects additional related data from a malware repository during the second phase. Finally, all collected data are stored in the database after passing through the last phase generating combined information in a structured format.

3.1. Phase 1: Parsing Indicators of Compromise. This phase starts with collecting cyber intelligence reports that analyze cyber incidents and malware interrelated APT campaigns and groups. For this, we obtain a list of papers from public repositories (APTnotes (https://github.com/aptnotes/data) APT & CyberCriminal Campaign Collection (https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections)) that provide publicly available articles and blog content related to malicious attacks, activities, and software associated with vendor-defined APT groups and/or tool sets. To maintain the usability of the dataset, we exclude the periodically published threat analysis reports from a list, integrating the analysis results from different APT groups that have no interrelation with each other. Therefore, it can be assumed that the data extracted during phases 1 and 2 are related to the same (or related) threat actors. We can use this property to set the ground truth of the data for analysis. This property and the dataset usability are explained in detail in Sections 4 and 5, respectively.

Next, Indicators of Compromise (IoCs) are extracted from the reports using a parser. We utilize a modified ioc_parser (https://github.com/armbues/ioc_parser) that extracts IoCs matched by predefined regular expressions such as the URL, host, IP address, email account, hashes (MD5, SHA1, and SHA256), common vulnerabilities and exposures (CVE), registry, file names ending with specific extensions, and the program database (PDB) path. Among the data obtained, the malware hash values are passed to the second phase for further data collection, and other values are passed to the last phase.

The IoC extraction performance is critically influenced by the performance of the parser. Therefore, other parsers can be chosen to increase the performance of this phase.

3.2. Phase 2: Collecting Analysis Data. Owing to the functional limitation of a parser, there may be remaining IoCs not extracted from the reports that can be found in malware analysis data. Moreover, we can obtain additional data from the analysis results that are not in the content of the reports. Notably, the valuable data, which cannot be expressed as a regular expression such as mutex, file mapping, code sign, and other strings, are only collectible from the malware analysis results.

To collect the malware analysis data, we use the malware repository service, malwares.com, operated by Saint Security

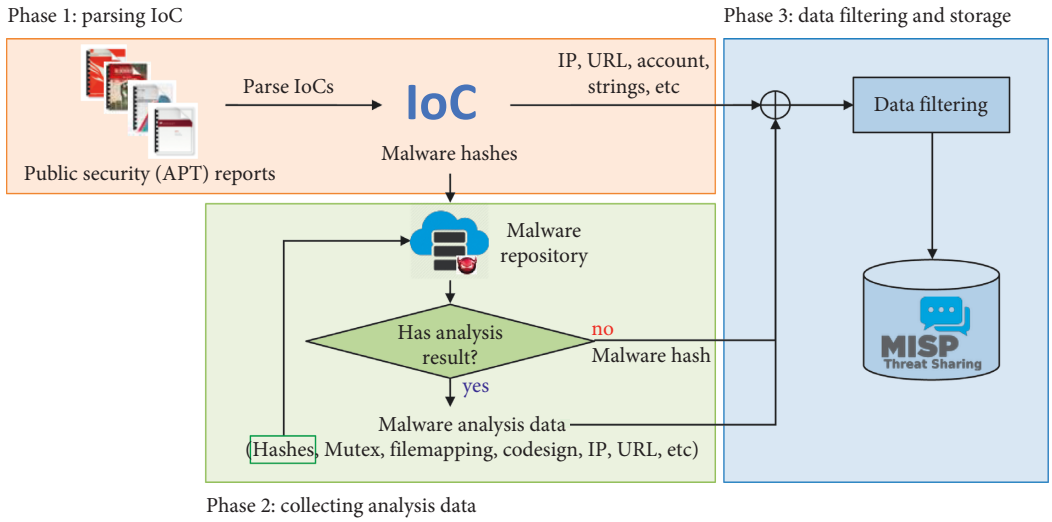


Figure 5: CTIMiner system architecture.

Inc., the first cloud-based malware analysis platform in South Korea. It possesses over 800 million malware samples and maintains a partnership with VirusTotal. If the malware analysis results are retrieved by querying the hash value, the data in the results, namely, hashes, URLs, IP addresses, PDB paths, code signs, file names, and other strings, are passed to the last phase; otherwise, the hash value itself is passed. We do not store malware samples in the database because of the possible occurrence of copyright concerns when it is publicly released. For the new hash values found from the results, the analysis data are gathered through the same procedure.

3.3. Phase 3: Data Filtering and Storage. The data collected from several sources may be redundant or noisy and can be filtered out during this phase. For example, some files are automatically generated by the operating system regardless of the intent of the malware creator when the malware is executed. We merge the repetitive data and remove noisy data during this phase. However, the tradeoff between false positives and false negatives needs to be considered for noise removal. The filtered data are stored in the MISP server, which provides an API to manage and export data in various structured formats.

Optionally, we categorize the data types composing the dataset and analyze their statistical characteristics during this phase, the results of which are presented in the next section.

3.4. System Processing Results of Phases 1 and 2. We ran this system on 612 collected APT reports published from January 2008 to June 2019, the numerical processing results of which are in Table 1. Among the 14,313 malware hashes extracted from the reports, we obtained analysis results for 68.1% of them from the malware repository. Among the analysis information, we found 450 new malware hashes that were not contained in the APT reports and added the analysis information to the dataset. The value of including the malware analysis data, in addition to the IoCs extracted from the reports, is described in the statistical analysis of the dataset provided in Section 4.2.

Table 1: Processing results of phases 1 and 2.

Types	No.	%
No. of reports	612	—
No. of data stored in the dataset	642,810	—
No. of extracted malware hashes from the reports	14,313	—
No. of analyzed malware	9,753	68.1
No. of additionally extracted malware	450	4.6

4. Dataset Descriptions

4.1. Dataset Structure and Data Types. The dataset is composed of several sets of events, and Figure 6 shows the relationship of one set of events, which is composed of two types of events, namely, one report event and several malware events. A report event includes the data extracted from the first phase described in Section 3, which parsed the texture IoCs from the APT reports. Malware events are created whenever malware hashes are detected, and it is possible to obtain their analyzed data during phase 2. These malware events and report events from which the malware hashes are originated can be grouped under the title of the report.

The data schema of an event is presented in Figure 7, and a short example of a set of events is shown in Figure 8. Because all malware events originating from one report include the same file name of the report, this can be used as the ground truth of the correlation analysis of the data. In addition, the malware compilation dates and the publication dates of the reports can be useful for a temporal analysis of the dataset. A sample application of the dataset for a correlation analysis using these dataset characteristics is demonstrated in Section 5.

The types of attributes stored in a dataset are the IP, URL, email address, date and time, CVE, file name, PDB path, digital code sign serial number, and other string data,

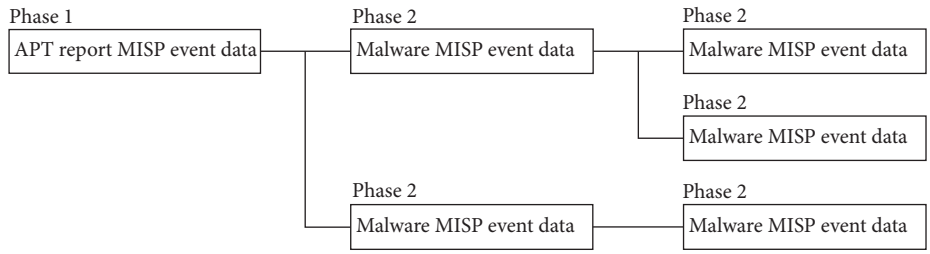


Figure 6: Relationship of a set of events.

```
<Event>
    <id> the event ID automatically assigned by MISP </id>
    <date> the publication date of a report or the time stamp of malwre </date>
    <info> the title of the report or the hash value of malware </info>
    <Attribute>
        <item>
            <category> the category of the data </category>
            <comment> the comment about the data </comment>
            <value> the data value </value>
            <type> the data type </type>
            <id> the attribute ID automatically assigned by MISP </id>
        </item>
        <item>
        .
        .
        .
        <item>
</Event>
```

Figure 7: Data schema of an event.

```
<Event>
    <id>2852</id>
    <date>2014-12-03</date>
    <info>Cylance_Operation_Cleaver_Report.pdf</info>
    <Attribute>
        <item>
            <category>External analysis</category>
            <comment />
            <value>zhcat.exe</value>
            <type>filename</type>
            <id>30996</id>
        </item>
        <item>
            <category>External analysis</category>
            <comment />
            <value>CVE-2010-0232</value>
            <type>vulnerability</type>
            <id>31056</id>
        </item>
        <item>
            <category>Network activity</category>
            <comment />
            <value>64.120.128.154</value>
            <type>ip-src</type>
            <id>30988</id>
        </item>
    </Attribute>
</Event>
```

```
<Event>
    <id>2741</id>
    <date>2014-12-03</date>
    <info>836ef6b06c5fd52ecc910a3e3408004a</info>
    <Attribute>
        <item>
            <category>External analysis</category>
            <comment>original_filename</comment>
            <value>zhCat.exe</value>
            <type>filename</type>
            <id>29863</id>
        </item>
        <item>
            <category>Network activity</category>
            <comment>TCP_undetected</comment>
            <value>1.224.181.13</value>
            <type>ip-src</type>
            <id>29864</id>
        </item>
        <item>
            <category>Payload installation</category>
            <comment />
            <value>723cdf97284e58a1672e031013620fe8d74e27f1</value>
            <type>sha1</type>
            <id>29861</id>
        </item>
        <item>
            <category>Artifacts dropped</category>
            <comment />
            <value>e:\Projects\Cleaver\trunk\MainModule\obj\Release\MainModule.pdb</value>
            <type>pdb</type>
            <id>29866</id>
        </item>
        <item>
            <category>Other</category>
            <comment />
            <value>Cylance_Operation_Cleaver_Report.pdf</value>
            <type>comment</type>
            <id>29867</id>
        </item>
    <Attribute>
</Event>
```

Figure 8: Example of a set of events.

including the author and title of the document. The amount of data, the report, and the malware events are shown in Table 2. Using the source codes that we publicly released, a dataset composed of the attribute types of interest can be created.

4.2. Data Categories and Statistics. We observed that the data collected from the reports and the malware analysis information are related to common cyber campaigns or threat actors, which can be categorized as shown in Figure 9.

The characteristics of each category are as follows:

① Data that can only be extracted by the parser belong in this category. The quality and quantity of this type of data depend highly on the contents of the reports and the functionality of the parser.

② Malware analysis data contained in reports but unable to be extracted by the parser belong in this category. The volume of this type of data shows how much malware analysis data can compensate for the limitation of the parser. In addition, the indicator of this category can be used to compare the quality of the analysis results from several malware repositories.

③ This category includes the data extracted by the parser as well as by the malware analysis results.

④ Some data related to campaigns or threat actors can be excluded in the APT reports owing to the low priority compared to other information or the analysis limitations of the authors. Such data found from malware analysis results belong to this category.

⑤ Noise data generated by the parser belong to this category. The functional limitation of the parser increases the portion of data in this category.

⑥ Data in this category are the noise generated from malware analysis information. It is difficult to distinguish between ④ and ⑥, but meaningless data generated by the runtime environment of malware belong to this category.

⑦ There are numerous data in the reports that are difficult for the parser or malware analysis information to obtain. Specifically, nontechnical information such as actors and groups of cyber campaigns mainly resides in this category. These data need to be extracted manually or by other supplemental methods.

⑧ This category is similar to ⑦ in the sense that neither data extraction methods can discover data in this category. Publishers can intentionally exclude the data, or may not even know about them. The volume of this category can be minimized by comparing several reports related to the same campaigns or threat actors, or by gathering multisource information such as HUMINT and SIGINT.

The statistical features of these dataset categories generated through phase 3 of the system are listed in Table 3. It is worth noting that 43% of the data come from malware analysis results (② and ④) and 26% are newly discovered data that are not contained in the reports (④). Comparing that the vast amount of data types in ② with the hash values, ④ consists of various types of data including code signs, IP addresses, and other string information valuable to identifying an incident.

5. Dataset Application

As mentioned previously, the objective of generating our dataset is to promote academic research related to CTI analysis. We propose three research topics applying a dataset and demonstrate one dataset application example in this section. Although it would be better if a novel analysis technique could be proposed, this is outside the scope of the present paper. The provided application example is the automatically generated correlation analysis results of the dataset using MISP.

5.1. Noise Removal. As described in Section 4, the dataset includes several types of noise, which makes a further data analysis difficult and causes erroneous results. The dataset contains noises owing to the malfunctions of the data extraction methods and the inclusion of less meaningful data. An effective noise removal technique should be able to consider the contextual necessities of the data among the entire dataset or sets of events. For example, the data contained in several sets of related events where there is little similarity of each event set is considered noise with a high probability because it increases the dissimilarity of the event sets correlated with the data.

5.2. Correlation Analysis. A proper usability of the dataset comes from finding the underlying relations among the data. Without any correlations, the dataset itself is nothing but a significant amount of scattered data that can only be used to search for the existence of certain items.

Because an event in the dataset is composed of several threat data, the correlations between events are determined by analyzing the relations among the threat data consisting of such events. A string-matching-based method provided by many commercial cyber intelligence services would be one way to find the relations of events. However, this simple method has several limitations. If two events contain attacker names such as "Bart Simpson" and "B. Simpson," a simple string-matching-based method will not find the relations between the events. Similarly, if the events include the URLs, "bartsimpson.com" and "bsimpson.net," the relations will also not be discovered. A string-similarity analysis and heuristics can be adopted to overcome such a limitation. Moreover, probabilistic approaches can improve an event-wise analysis when considering the relations among sets of data of the events.

5.3. Temporal Analysis. Understanding the history of cyber campaigns by adversaries is crucial, not only to defend against current incidents and presume the underlying intent, but also to draw the direction of adversarial activities from

Table 2: Number of data for each type.

Year	Data types											Report	Malware
	Hash	IP	URL	Email	Date, time	CVE	File name	PDB	Code sign	Others	Total		
2008	0	3	171	0	0	0	17	0	0	0	191	2	0
2009	2	7	84	2	0	0	10	0	0	0	105	2	0
2010	223	79	280	14	32	2	213	0	0	0	800	7	32
2011	1,440	412	478	17	319	7	713	2	38	25	3,340	14	319
2012	2,240	433	637	46	465	30	828	2	43	7	4,524	22	465
2013	8,329	2,505	3,032	599	1,798	45	3,003	97	802	61	19,571	47	1,798
2014	5,614	5,484	3,282	476	1,116	83	2,804	22	438	28	18,842	100	1,116
2015	6,801	2,752	2,658	334	1,554	48	3,077	28	206	34	17,258	78	1,554
2016	8,001	525,020	3,449	235	1,833	81	4,873	43	154	14	543,703	79	1,974
2017	4,343	3,316	3,582	534	935	49	2,780	13	99	9	15,660	72	1,017
2018	3,900	3,296	2,582	229	0	74	2,660	34	404	31	13,210	125	1,300
2019 (–Jun.)	2,046	719	1,439	194	0	51	1,110	9	36	2	5,606	64	628
Total	42,939	544,026	21,674	2,680	8,052	470	22,088	250	2,220	211	642,810	612	10,203

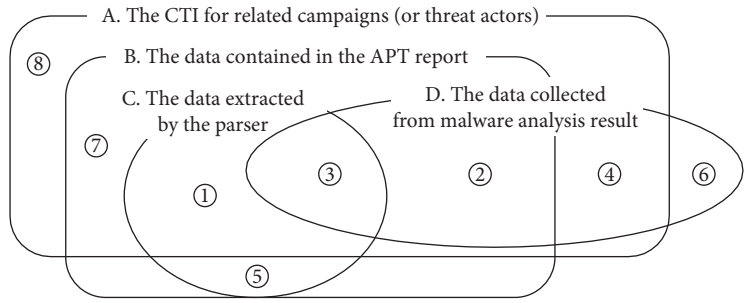


Figure 9: Data categories of the dataset.

Table 3: Percentage of data in each category.

Category	①	②	③	④
%	46	17	11	26

the big picture. Furthermore, the tactics, techniques, and procedures identified from the campaigns through a temporal analysis can be used to characterize the behavior of the adversarial groups. Therefore, the characteristics can be applied as a feature for a correlation analysis of the sequences of events.

5.4. Example Dataset Application. The proposed dataset can be used for a correlation analysis of cyber incidences. The cyber threat actor group retrieving the correlation in the example is the Lazarus group, which has been suspected to have conducted many major cyber campaigns, including the following:

(i) The Sony Pictures Entertainment attack (2014)

(ii) A bank heist including the Bangladesh Bank (2016)

(iii) The worldwide WannaCry ransomware distribution (2017)

We conducted a correlation analysis of a dataset collected by CTIMiner with help from the MISP correlation graph shown in Figure 10.

The starting point of the correlation analysis is a security report on "Lazarus' False Flag Malware [32]," marked as ⓐ. As mentioned in the report, the Lazarus group was involved in a polish banks heist, and the corresponding report of which is [33] marked as ⓑ. The datawise correlation of incidents can be found in Figure 10. The data in ①, which are extracted from the reports and from malware analysis results, correlate ⓐ and ⓑ, and the others in ② link ⓐ to ⓒ, which is another report from BAE systems regarding the Lazarus group. Therefore, through ⓐ, ⓑ, and ⓒ may have a correlation.

Although this paper does not intend to propose CTI analysis techniques, by applying a previously proposed dataset application, we can deduce practical lessons on how this dataset can be used for CTI generation in this example. A CTI analysis algorithm is basically able to find the connectivity of the data extracted from the same APT report. In advance, the algorithm can correlate the reports that analyze the same attributes and campaigns. A CTI analysis algorithm should eventually aim to generate actionable intelligence allowing patterns of attack to be determined as a means to predicting the intent of the attackers and to prepare against similar attacks.

Kim et al. proposed an event-centric correlation analysis approach to assist in generating such CTI. They suggested a novel concept and a construction algorithm that expresses the similarities among threat events and temporal characteristics in a graphical structure [12]. To use our CTI dataset for an advanced analysis, successive studies should be conducted.

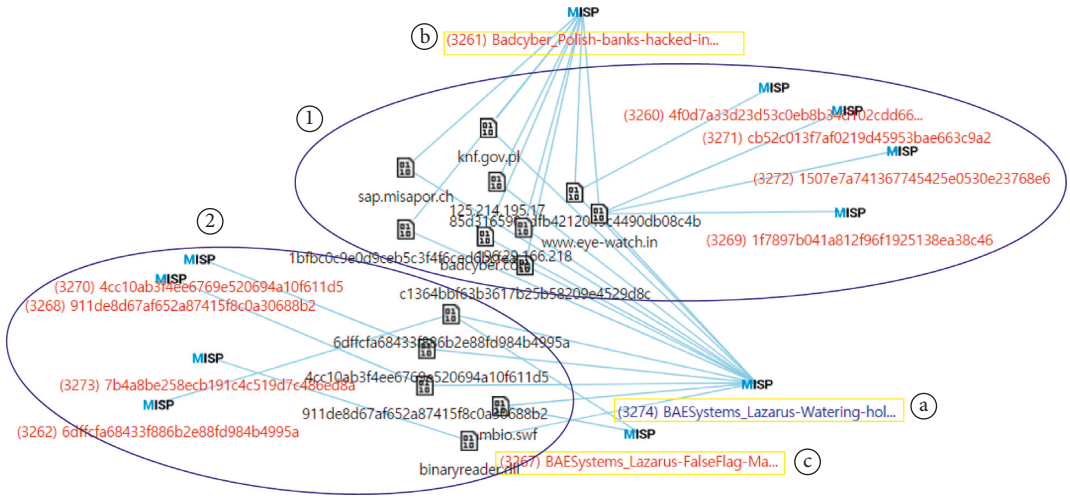


Figure 10: Sample application of the dataset to a correlation analysis.

6. Conclusion

Owing to the prevalence of cyber threats and a rapid increase in the amount of data collected, researchers are developing techniques for the different intelligence processes to be actively conducted. However, compared to other intelligence processing steps, studies have been undertaken limitedly for the analysis and production step that requires the real CTI dataset for the analysis. We pointed out that dataset unavailability is the main reason suppressing vitalization of the research despite many interests. To address the problem, we proposed CTIMiner system that generates the dataset consisted of the data contained in security reports and supplemented with malware analysis data related to the reports. After categorizing the types of data collected from the system, we provided the statistical feature of the dataset. To show the usability and applicability of the dataset, we proposed several research topics possible to be conducted using the dataset and demonstrated the correlation analysis result for an event in the dataset.

Our future research direction is to develop and enhance the proposed analysis technique using the dataset on top of the CTI correlation analysis framework [12]. By releasing this dataset to the public, we believe it can promote the threat analysis research to generate CTI.

Data Availability

The source codes of CTIMiner system and the generated dataset described in this paper are available to the public. These are accessible at our GitHub repository (https://github.com/dgkim0803/CTIMiner). Using the source codes, security reports, and MISP, one can generate a dataset composed of the data types that one has interested in.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported under the framework of international cooperation program managed by the National Research Foundation of Korea (No. 2017K1A3A1A17092614).

References

[1] R. McMillan, *Definition: Threat Intelligence*, Gartner, Stamford, CT, USA, 2013.

[2] *Senate Resolution 754, Cybersecurity Information Sharing Act of 2015 (CISA)*, Council on Foreign Relations, New York, NY, USA, 2015.

[3] *Cybersecurity Alliance for Mutual Progress (CAMP)*, 2016, https://www.cybersec-alliance.org.

[4] Olympic destroyer is here to trick the industry," *GReAT*, 2018, https://securelist.com/olympicdestroyer-is-here-to-trick-theindustry/84295/.

[5] *FACT SHEET: President Xi Jinping's State Visit to the United States*, The White House, Washington, DC, USA, 2015.

[6] C. Johnson and D. Waltermire, *Special Publication 800-150: Guide to Cyber Threat Information Sharing*, National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA, 2014.

[7] "The MITRE cybersecurity standards," 2013, https://www.mitre.org/capabilities/cybersecurity/overview/cybersecurityresources/standards.

[8] J. Rosenberg, "2018 winter cyber olympics: code similarities with cyber attacks in Pyeongchang," 2018, http://www.intezer.com/2018-winter-cyber-olympics-code-similarities-cyber-attacks-pyeongchang.

[9] M. L. P. Rascagneres, "Who wasn't responsible for olympic destroyer?," 2018, https://blog.talosintelligence.com/2018/02/who-wasnt-responsible-for-olympic.html.

[10] J. A. Guerrero-Saade, P. Moriuchi, and G. Lesnewich, "Targeting of olympic games it infrastructure remains unattributed," 2018, https://www.recordedfuture.com/olympic-destroyer-malware.

[11] *Joint Publication 2-0: Joint Intelligence*, U.S. Joint Chiefs of Staff, USA, 2013.

[12] D. Kim, J. Woo, and H. K. Kim, ""I know what you did before": General Framework for Correlation Analysis of Cyber Threat Incidents," in *Proceedings of the IEEE 35th*

International Conference on Military Communications, Baltimore, MD, USA, 2016.

[13] S. Goel, "Cyberwarfare," *Communications of the ACM*, vol. 54, no. 8, pp. 132–140, 2011.

[14] V. Benjamin, W. Li, T. Holt, and H. Chen, "Exploring threats and vulnerabilities in hacker web: Forums, IRC and carding shops," in *Proceedings of the IEEE International Conference on Intelligence and Security Informatics*, Baltimore, MD, USA, May 2015.

[15] C. Fachkh and M. Debbabi, "Darknet as a source of cyber intelligence: survey, taxonomy, and characterization," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 2, pp. 1197–1227, 2016.

[16] *Information Marketplace for Policy and Analysis of Cyber-risk & Trust (IMPACT)*, U.S. Department of Homeland Security, Washington, DC, USA, 2014, https://www.impactcybertrust.org.

[17] *DHS/S&T/PIA-006 Protected Repository for the Defense of Infrastructure against Cyber Threats*, Washington, DC, USA, 2011, https://www.dhs.gov/publication/dhsstpia-006-protected-repository-defense-infrastructure-against-cyber-threats.

[18] "Standardizing cyber threat intelligence information with the Structured Threat Information eXpression (STIX)," The MITRE Corporation, McLean, VA, USA, 2012.

[19] "Sophisticated Indicators for the Modern Threat Landscape: An Introduction to OpenIOC," MANDIANT, Alexandria, VA, USA, 2011.

[20] X. Liao, K. Yuan, X. Wang, Z. Li, L. Xing, and R. Beyah, "Acing the IOC game: toward automatic discovery and analysis of open-source cyber threat intelligence," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, Vienna, Austria, October 2016.

[21] A. Kornmaie and F. Jaouën, "Beyond technical data-a more comprehensive situational awareness fed by available intelligence information," in *Proceedings of the 6th International Conference On Cyber Conflict*, Tallinn, Estonia, June 2014.

[22] Z. S. A. P. T. K. Ajay Modi, Z. Zhao, A. Doup, and P. Black, "Towards automated threat intelligence fusion," in *Proceedings of the IEEE 2nd International Conference on Collaboration and Internet Computing*, Pittsburgh, PA, USA, November 2016.

[23] S. Truvé, "Temporal analytics for predictive cyber threat intelligence," in *Proceedings of the 25th International Conference Com- panion on World Wide Web*, Montreal, Canada, April 2016.

[24] S. Qamar, Z. Anwar, M. A. Rahman, E. Al-Shaer, and B.-T. Chu, "Data-driven analytics for cyber-threat intelligence and information sharing," *Computers & Security*, vol. 67, pp. 35–58, 2017.

[25] *Special Publication 800-150: Guide to Cyber Threat Information Sharing*, National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA, 2014.

[26] J. Connolly, M. Davidson, M. Richard, and C. Skorupka, *The Trusted Automated eXchange of Indicator Information (TAXII)*, The MITRE Corporation, McLean, VA, USA, 2012.

[27] *Malware Information Sharing Platform (MISP)*, https://www.misp-project.org/.

[28] *The MANTIS Cyber Threat Intelligence Management Framework*, https://github.com/siemens/django-mantis.

[29] Collective Intelligence Framework (CIF)," *CSIRT Gadgets*, https://csirtgadgets.com/collective-intelligence-framework.

[30] S. W. Seo, J. H. Kim, D. R. Lee, and H. K. Kim, "C-TAS Ecosystem for Cyber Threat Analysis & Sharing in Korea," in *Proceedings of the OASIS Borderless Cyber and FIRST Technical Symposium*, Prague, Czech Republic, December 2017.

[31] W. Zha and G. White, "A Collaborative Information Sharing Framework for Community Cyber Security," in *Proceedings of the IEEE International Conference on Technologies for Homeland Security*, Waltham, MA, USA, November 2012.

[32] S. Shevchenko, *Lazarus' False Flag Malware*, 2017, http://baesystemsai.blogspot.kr/2017/02/lazarus-false-flagmalware.html.

[33] "Several Polish banks hacked, information stolen by unknown attackers," BadCyber, 2017, https://badcyber.com/severalpolish-banks-hacked-information-stolen-by-unknown-attackers/.

Hindawi
Security and Communication Networks
Volume 2019, Article ID 1982168, 11 pages
https://doi.org/10.1155/2019/1982168

Research Article

Power Grid Estimation Using Electric Network Frequency Signals

Woorim Bang and Ji Won Yoon

Graduate School of Information Security, Korea University, Seoul, Republic of Korea

Correspondence should be addressed to Ji Won Yoon; jiwon_yoon@korea.ac.kr

Received 7 April 2019; Revised 12 July 2019; Accepted 25 August 2019; Published 24 September 2019

Guest Editor: Sebastian Schrittwieser

Copyright © 2019 Woorim Bang and Ji Won Yoon. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The electric network frequency (ENF) has a statistical uniqueness according to time and location. The ENF signal is always slightly fluctuating for the load balance of the power grid around the fundamental frequency. The ENF signals can be obtained from the power line using a frequency disturbance recorder (FDR). The ENF signal can also be extracted from video files or audio files because the ENF signal is also saved due to the influence of the electromagnetic field when video files or audio files are recorded. In this paper, we propose a method to find power grid from ENF signals collected from various time and area. We analyzed ENF signals from the distribution level of the power system and online uploaded video files. Moreover, a hybrid feature extraction approach, which employs several features, is proposed to infer the location of the signal belongs regardless of the time that the signal was collected. Employing our suggested feature extraction methods, the signal which extracted from the power line can be classified 95.21% and 99.07% correctly when ENF signals have 480 and 1920 data points, respectively. In the case of ENF signals extracted from multimedia, the accuracy varies greatly according to the recorded environment such as network status and microphone quality. When constructing a feature vector from 120 data points of ENF signals, we could identify the power grid had an average of 94.17% accuracy from multimedia.

1. Introduction

A supply frequency of electrical power in power distribution networks is called electric network frequency (ENF). In most countries, the ENF has a nominal value of 50 or 60 Hz. It is essential to maintain a constant frequency value to prevent a total collapse of the electric system and to keep the grid stable. In order to balance between aggregate generation and aggregate demand across the interconnection, the ENF signals have small variations every moment [1]. This change is a normal phenomenon for stable operation of the power grid and is acceptable if the value of ENF signal is less than the threshold. Because of these characteristics of the ENF, the ENF has uniqueness according to time and location.

There are two main methods for obtaining the ENF signals. One is to collect the ENF signals utilizing special equipment such as frequency disturbance recorders (FDRs). The other is to extract them from multimedia. When recording audio files or video files, the ENF signals are also being stored together because of the electromagnetic influences from the power line [2]. In other words, the ENF signals can be included in multimedia. Thus, the ENF signals in video files or audio files can be applied to digital forensics to determine whether the file is modified or not [3, 4]. In order to demonstrate the authenticity of the file, reference database is required to store the value of the ENF signals extracted from the same time and the same area as the file is required. Since most forensics is done after a series of events, a database of the ENF signals requires to be established for almost all regions and times for the investigation. This makes it difficult to use the ENF signal for digital forensics. If we can suggest which region an arbitrary ENF signal belongs to, it will increase the value of applying the ENF signal to the digital forensics.

In most countries, the location privacy is regarded as one of the important personal information. Numerous people around the world freely upload and download numerous photos, audio files, and video files through social network services, video sharing web services, and live webcam services such as Facebook [5], Instagram [6], YouTube [7],

Earthcam [8], Skyline Webcams [9], and explore [10]. To comply with personal privacy regulations, web service providers remove GPS information from uploaded photo or video file's metadata before posting on online or obtain user's agreement to release their location information. However, there is a research that not only the location information exists in the metadata of audio files or video files but also the location could be inferred from ENF signals in audio files or video files [11, 12]. Kim et al. [12] conducted a study on the construction of a national-scale ENF map using multimedia recorded in several areas. Jeon et al. [11] suggested intragrid location estimation of audio files and video files from Voice over Internet Protocol (VoIP) applications and streaming services using ENF map. Both studies [11, 12] require ENF signals collected at the same for inferring location.

In this paper, we introduce the model that extracts features from ENF signals and identifies the power grid where ENF signals are extracted from. Using our proposed method, we can estimate the location even if ENF signals are not obtained from close time to the train dataset because we extract features by power grids. We evaluated our model on three power grids in the United States and the Eastern, the Western, and the Texas power grid. We collected ENF signals from these power grids. We constructed two types of ENF signal datasets. The first is composed of ENF signals at the distribution level of the power systems. Another is extracted from video files and audio files provided by worldwide live webcam services, such as Skyline Webcam [9], Earthcam [8], and explore [10] and video sharing web service YouTube [7] on the Internet.

To extract enough features to classify the power grid, we mainly merge features extracted from three different methods which consist of the autoregressive coefficients using the Yule–Walker method, the Shannon entropy on terminal nodes of maximal overlap discrete wavelet packet transform, and the multiscale variance of maximal overlap discrete wavelet transform. Finally, a feature vector is constructed by adding the variance of ENF signals into the merged feature from three main feature extraction algorithms. The support vector machine (SVM) and XGBoost [13] are compared as a method of classifying the power grid using extracted features. We finally evaluated the experimental results of our proposed method depending on the number of ENF signals. Through our proposed method, we have been able to identify power grids with high accuracy.

Our main contributions are summarized as follows:

(i) We can estimate the power grid of ENF signals extracted from the power line and online multimedia. By analyzing ENF signals, the power grid can be efficiently identified without reference data which are composed of ENF signals extracted from the same time and the same region of recorded files.

(ii) Using our proposed feature extraction algorithm, we can perform analysis on much less data than the number of raw signals. We propose to construct a feature vector from the three main feature extraction methods.

(iii) We studied the factors that influence the estimation of the power grid through ENF signals. First of all, we have investigated the power grid identification accuracy according to the number of ENF signals. In addition, we analyzed the power grid estimation accuracy conducted on the harmonic components used in extracting ENF signals from online multimedia.

The structure of this paper is as follows. In Section 2, we introduce the related work of estimating location using ENF signals. Moreover, we present a related research that extracts ENF signals from a multimedia file with sound recorded such as audio files or video files. Section 3 shows the two main background knowledge required in this paper. In Section 4, we explain the two types of the ENF signal dataset used in our proposed method. We propose our algorithm to estimate the power grid using ENF signals in Section 5. We evaluate our proposed technique using the accuracy and F1 score in Section 6. In Section 7, we discuss the importance of our experiments and the limitations of our work. Finally, we conclude this paper in Section 8.

2. Related Work

In this section, we present the related research of location estimation using ENF signals and the study of extracting ENF signals from multimedia.

2.1. Location Estimation Using ENF Signals. Over the past few years, researches have been consistently proposed to classify ENF signals by location [2, 14–16]. Yao et al. [15] suggested the wavelet-based signature extraction and feed-forward artificial neural network-based machine learning. They required a minimum length of 15 minutes of uninterrupted frequency data (9000 continuous data points) using the FNET/GridEye system. In real-world environments, it is not easy to gather data at every 0.1 second intervals for 15 minutes due to temporal equipment failures, network failures, or server load. When we collect data using the same system, the FNET/GridEye system, data loss often occurred. In Section 4.1, we introduce a data preprocessing technique that can use data even if it involves some data loss.

Hajj-Ahmad et al. [2, 14] and Šarić et al. [16] suggested a classification method for ENF signals using several feature extraction algorithms. Šarić et al. [16] used between 305 and 480 minutes of power signals and 60 minutes of audio recording to classify the power grid. Hajj-Ahmad et al. [2, 14] suggested analyzing 96 samples of ENF signals from power signals and audio recordings. In previous studies [2, 14, 16], there was a lack of research on estimation power grid accuracy according to the number of ENF data points. In Section 6, we evaluated the power grid identification accuracy for various ENF signal length.

2.2. Extraction of ENF Signals from Multimedia. Kim et al. [12] and Hajj-Ahmad et al. [17] have studied methods for extracting ENF signals from multimedia recordings. Hajj-

Ahmad et al. [17] have tested with extracting ENF signals from recordings using MUltiple SIgnal Classification (MUSIC) and Estimation of Signal Parameters via Rotational Invariance Techniques (ESPRIT) methods. Kim et al. [12] used the Quadratically Interpolated Fast-Fourier Transform (QIFFT) method [18] and multitone harmonics technique to obtain ENF signals from multimedia data. In this paper, we extracted the ENF signals using QIFFT and multitone harmonics in the video files uploaded on the Internet. Moreover, we examined the relationship between the number of harmonic frequency bands which are used to extract ENF signals and power grid identification accuracy.

3. Background

In this section, we describe existing techniques that identify the power grids and methods for extracting ENF signals from online multimedia.

3.1. Electric Network Frequency (ENF). The electric network frequency (ENF) is the supply frequency of power grids. The ENF signals exist in nominal value at either 50 or 60 Hz depending on geographic location [19]. If the value of ENF signals exceeds the threshold value, it causes all equipment operate using electricity. Therefore, the value of the supply frequency should always be constant. However, the ENF signals fluctuate a little over around the dominant value because of the imbalance of the supply and demand of power grids. Thus, the ENF signals have uniqueness by time and location.

3.2. Extraction of ENF Signals from Online Multimedia. Kim et al. [12] applied multitone harmonics method and QIFFT (Quadratic Interpolated Fast-Fourier Transform) method to the audio of multimedia data to obtain ENF signals. When extracting ENF signals from multimedia data, the most important task is how to reduce a noise. Therefore, they applied multitone harmonics as a technique to reduce noise and obtain more accurate ENF signals. The ENF signal is generally present at 50/60HZ as a fundamental frequency as well as in harmonic frequency bands which are integer multiplication of fundamental frequency (Figure 1). Kim et al. [12] selectively used harmonic frequency bands and applied QIFFT to extract the maximum value of ENF signals from each window.

4. Dataset Description

This section explains the two main ENF datasets that were used to evaluate the ENF signal classification. In general, there are two approaches to acquire ENF signals. One is obtained from the distribution level of the power system. The other is extracted from audio files and video files.

4.1. ENF Signal Extracted from Power Grid Distribution Level. We acquired the ENF signal extracted from the distribution level of the power system using the FNET/GridEye system [20]. A wide-area frequency monitoring network (FNET)

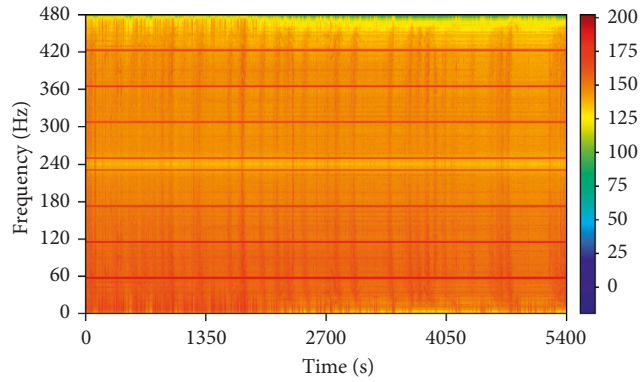


Figure 1: ENF signals at fundamental frequency (60 Hz) and harmonic frequency bands (around 120, 180, 240, 300, 360, 420, and 480 Hz) in multimedia file.

consists of the frequency disturbance recorder (FDR) data collected from the three interconnections in North America [21] and the Eastern power grid, the Western power grid, and the Texas power grid. The FDRs, members of the Phasor Measurement Unit (PMU) family, measure the voltage phase angle, amplitude, and frequency from a single-phase voltage source at 120 V distribution level with GPS synchronized. 129 units of FDR are installed in the Eastern Interconnection, 7 units in the Texas Interconnection, and 45 units in the Western Interconnection in the United States. The FNET had provided a web service which dynamically updates the frequency data every 4 seconds received from the FDRs until April 30, 2018 [22]. We collected electric network frequency data for more than 180 United States regions provided by FNET web service every 4 seconds for 49 days.

Despite of the high accuracy of the FDR (± 0.0005 Hz) [21], we find that there often exist invalid or missing data in the collected data due to hardware failures such as GPS signal loss, aging of equipment or network interruption and web server failure. Therefore, data preprocessing is required for enhancing the accuracy of the experiment because the time interval for collecting data during 49 days is not always constant at 4 seconds. First of all, we selectively choose areas which collected over 99% for data accuracy; 52 regions in the Eastern power grid, 19 regions in the Western power grid and 2 regions in the Texas power grid. For each selected region, linear interpolation is employed every second based on the collected data to fill the missing data. After linear interpolation, we subsampled every 15 seconds for 49 days not only to reduce the features of the data but also to avoid distortion due to data loss. Accordingly, it means that there are 4 data points per minute and 5760 data points per day for each FDR installed area.

4.2. ENF Signal Extracted from Online Multimedia. To extract ENF signals from multimedia data, we gathered audio files and video files from online web services. To classify the location, we collected videos from four web services: Skyline Webcams [9], EarthCam [8], explore [10], and YouTube [7]. Skyline Webcams [9], EarthCam [8], and explore [10] are worldwide live webcam services which stably provide videos

enough to extract the ENF signals as well as including location information about videos. In addition, we collected multimedia data from YouTube, which is used by many users worldwide for video sharing. In YouTube which does not provide accurate location information, we selectively gathered videos such as concerts, festivals, and studio broadcasts of local broadcasting stations which are certain of the location where the video was taken. When we gathered multimedia, we only need data from the sound to obtain the ENF signals by analyzing the frequency bands. Therefore, we set the sampling rate to 1000 Hz and stored only sounds to utilize storage space efficiently.

In contrast to the ENF signals obtained at the distribution levels, which only have the fundamental frequency (50 or 60 Hz), multimedia files contain a wide range of frequencies as well as fundamental frequency. Thus, unlike the ENF signals collected at the distribution level, additional steps are required to obtain the ENF signals from the multimedia. We used multitone harmonics method and QIFFT method to extract ENF signals from videos proposed by Kim et al. [12]. Some files have ENF signals in either fundamental frequency or harmonics frequency bands. However, some of the files are included in neither the fundamental frequency nor the harmonic frequency bands. For a file including ENF signals at least one frequency band, the number of frequency bands used to extract the ENF signal is shown in Table 1. When ENF signals are extracted using all harmonic frequencies including fundamental frequency, they are extracted using a total of 8 frequency bands. When the sampling rate is 1000 Hz and the window size for QIFFT is set to 296000, about 80% of the 374 files have ENF signals in all frequency bands. The distribution of the number of harmonic bands used and the harmonic band frequency used is shown in Figure 2. The extraction of the ENF signal using eight harmonic band frequencies is much more than the extraction of the ENF signal using two to seven harmonic band frequencies. Thus, it displays the employed frequency information corresponding to using 2 ~ 7 frequency bands from the multimedia files to extract ENF signals.

We conducted experiments on 374 video files which were taken in the United States of America and include ENF signals. The videos were collected for about two years, and the length of the video recorded is varied from 15 minutes to over 4 hours (see details of the videos in Table 2).

Table 1: Number of frequency bands employed to extract ENF signals in videos (the videos are limited to including ENF signals).

Number of frequency bands	1	2	3	4	5	6	7	8
Number of multimedia	0	8	7	12	9	12	28	298

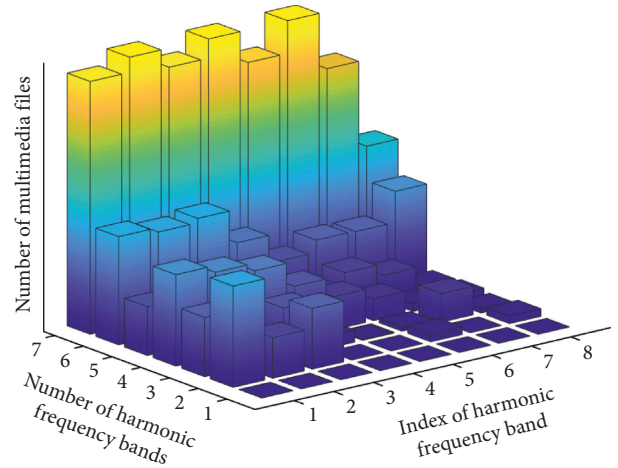


Figure 2: The distribution of the number of harmonic bands and the information of extracted harmonic band frequency used to restore the ENF signals from the multimedia files.

Table 2: Number of videos by recorded length.

Time (hour)	0.25 ~	0.5 ~	1 ~	2 ~	3 ~	4 ~
Number of multimedia	374	199	20	3	2	1

5. Proposed Approaches

In this section, we introduce how to construct a feature vector which are composed of three different feature extraction methods for the ENF signals and a classification method to the power grid.

5.1. Feature Extraction on ENF Signals. We present three different methods for extracting features from ENF signals. To obtain the envelope of the signal spectrum feature, we employ the autoregressive model in Section 5.1.1. An approach for extracting time-invariant properties from a signal is given in Section 5.1.2. The maximal overlap discrete wavelet packet transform was used to determine the local characteristics in Section 5.1.3. Using the results obtained by three feature extraction methods from ENF signals, we were able to construct a feature vector.

5.1.1. Autoregressive (AR) Coefficients. The basic idea of the autoregressive model is that the value at time t, x_t, depends linearly on its own p previous values, $x_{t-1}, x_{t-2}, \ldots, x_{t-p}$, with a white noise ε_t at time t. The notation AR(p) indicates an autoregressive model of order p. AR(p) defined as

$$x_t = \sum_{i=1}^{p} \varphi_i x_{t-i} + \varepsilon_t, \tag{1}$$

where φ_i is a parameter of the model and ε_t is a white noise. To calculate the parameters, we used the Yule–Walker equation which is also called the autocorrelation method. The Yule–Walker equation for the autoregressive model is based on minimizing the forward prediction error in the least-square sense. The Yule–Walker equation can be represented by a matrix form as follows:

$$\begin{pmatrix} r_0 & r_1 & r_2 & \cdots & r_{p-1} \\ r_1 & r_0 & r_1 & \cdots & r_{p-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{p-1} & r_{p-2} & r_{p-3} & \cdots & r_0 \end{pmatrix} \begin{pmatrix} \varphi_1 \\ \varphi_2 \\ \vdots \\ \varphi_p \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_p \end{pmatrix}, \tag{2}$$

where r_d is the autocorrelation coefficient at delay d and the diagonal value r_0 sets to 1. As a result, the Yule–Walker equation gives the same number of parameters as the order of the autoregressive model. In our experiment, the order of the autoregressive model, p, was set to 12. Therefore, 12 features can be obtained per ENF segment.

5.1.2. Multiscale Variance of Maximal Overlap Discrete Wavelet Transform. In order to find the time-invariant property, we used the maximal overlap discrete wavelet transform (MODWT) decomposition which retains all possible times at each time scale [23]. Similar to the discrete wavelet transform (DWT), the MODWT can be used in a multiresolution analysis. Therefore, we perform the MODWT of the signals using the Daubechies wavelet with 2 of vanishing moments (db2) down to the maximum level. The wavelet and scaling filter coefficients at level j result from the inverse discrete Fourier transform (DFT) of a product of DFTs. The DFTs in the product are the signal's DFT and the DFT of the j^{th} level wavelet or scaling filter. Let MODWT wavelet and scaling filters of the length N DFTs be H_k and G_k. The j^{th} level wavelet filter is defined as [24]

$$\frac{1}{N}\sum_{k=0}^{N-1} H_{j,k} e^{(i2\pi nk)/N}, \tag{3}$$

where

$$H_{j,k} = H_{2^{j-1}k \bmod N} \prod_{m=0}^{j-2} G_{2^m k \bmod N}. \tag{4}$$

The j^{th} level scaling filter is defined by [24]

$$\frac{1}{N}\sum_{k=0}^{N-1} G_{j,k} e^{(i2\pi nk)/N}, \tag{5}$$

where

$$G_{j,k} = \prod_{m=0}^{j-1} G_{2^m k \bmod N}. \tag{6}$$

After MODWT, we got the unbiased estimates of the wavelet variance by scale as features. The wavelet variance is computed as follows:

$$\widehat{\nu}_j^2 = \frac{1}{N_j}\sum_{t=0}^{N_j-1} W_{j,t}^2, \tag{7}$$

where N_j is the number of coefficients at level j and $W_{j,k}$ is the wavelet coefficients of level j at k^{th} node [23].

In order to obtain the unbiased estimates of the wavelet variance by scale as features, we have to perform the maximal overlap discrete wavelet transform (MODWT) of the signals. Figure 3 displays the MODWT of the signals using the Daubechies wavelet with 2 of vanishing moments down to the maximum level when the 12-hour ENF signals are given. After performing MODWT, we estimated the wavelet variance. In our experiment, we could obtain less than $[\log_2 N + 1]$ features when the ENF segment consists of N data samples.

5.1.3. Shannon Entropy on the Terminal Nodes of Maximal Overlap Discrete Wavelet Packet Transform. The maximal overlap discrete wavelet packet transform (MODWPT) does not perform downsampling unlike discrete wavelet transform and has time-invariant properties and shifting invariant properties. Moreover, the coefficients of the MODWPT can include the local characteristics information of the signal. In spite of these properties, it is difficult to directly use it as a feature for classification because of the large number of coefficients [25]. In order to analyze the relevance of the coefficients, we employed the Shannon entropy. As a result, we calculate the Shannon entropy from the terminal node of the MODWPT as a feature.

We denote the ENF time series to be MODWPT, $x_t\,(t = 0, 1, \ldots, N-1)$, by $\mathbf{x}$, where N is the length of the series. The sequence of MODWPT coefficients at level $(j = 0, 1, \ldots, J)$ and frequency index $(n = 0, 1, \ldots, 2^j - 1)$ represents as $W_{j,n} = \{W_{j,n,t}, t = 0, 1, \ldots, N-1\}$. To compute the maximal overlap wavelet packet coefficients for levels, we can recursively filter the wavelet packet coefficients at the previous stage. Let $\mathbf{W}_{0,0} \equiv \mathbf{x}$ and given the series $W_{j-1,n/2,t}$ of length N, we can derive $W_{j,n,t}$ using [26]

$$\begin{aligned} W_{j,n,t} &\equiv \sum_{l=0}^{L-1} \widetilde{f}_{n,l} W_{j-1,n/2,(t-2^{j-1}l) \bmod N}, \\ \widetilde{f}_{n,l} &= \begin{cases} \widetilde{g}_l, & \text{if } n \bmod 4 = 0 \text{ or } 3, \\ \widetilde{h}_l, & \text{if } n \bmod 4 = 1 \text{ or } 2, \end{cases} \end{aligned} \tag{8}$$

where $L \le N$, $\widetilde{g}_l = g_l/\sqrt{2}$, and $\widetilde{h}_l = h_l/\sqrt{2}$. The scaling (low-pass) filter is denoted as g_l and the wavelet (high-pass) filter is represented as h_l.

The total wavelet energy of the n^{th} node at level j is defined as

$$E_{j,n} = \sum_{t=0}^{N-1} \left\| W_{j,N,t} \right\|^2. \tag{9}$$

The probability of the t^{th} coefficient represents as

$$p_{j,n,k} = \frac{\left\| W_{j,N,t} \right\| 2}{E_{j,n}}. \tag{10}$$

That is, $p_{j,n,k}$ are the normalized squares of the wavelet packet coefficients in the n^{th} node at level j. Finally, the Shannon entropy which is an measure of information entropy is given by

$$\text{SE}_{j,n} = -\sum_{t=0}^{N-1} p_{j,n,k} \times \log p_{j,n,t}. \tag{11}$$

In order to calculate the Shannon entropy, we perform a level 2 decomposition of the raw ENF signals using the MODWPT. By decomposing the MODWPT to level 2, there are four nodes at level 2: $W_{2,0}$, $W_{2,1}$, $W_{2,2}$, and $W_{2,3}$. The results of MODWPT for one ENF segment can be seen in Figure 4. After performing MODWPT up to level 2, Shannon entropy was calculated for each of the four terminal nodes ($W_{2,0}$, $W_{2,1}$, $W_{2,2}$, and $W_{2,3}$).

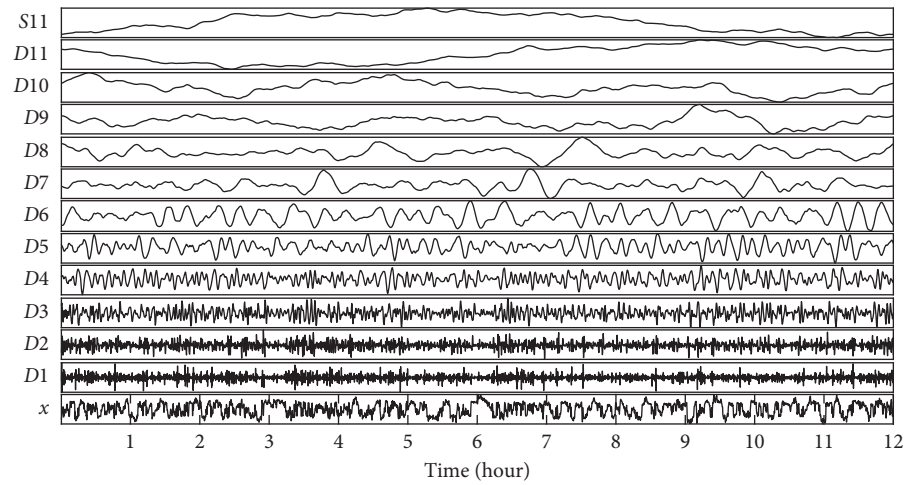


Figure 3: The maximal overlap discrete wavelet transform decomposition of the raw ENF signals during 12 hours (**x**). The different frequency components of the raw ENF signals displayed as the orthogonal components ($D1, D2, \ldots, D11$). The smoothed components are plotted in $S11$.

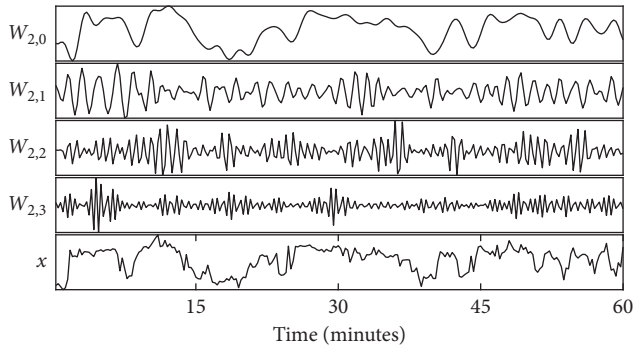


Figure 4: Raw ENF signal, **x**, and two-level maximal overlap discrete wavelet packet transform for the raw signals.

5.1.4. Feature Vector Configuration through Integration of Feature Extraction Methods. The ENF signal fluctuates randomly over time around 60 Hz, and it is difficult to predict the pattern [4]. Therefore, it is hard to obtain enough features to classify grid in ENF signals by using only one feature extraction technique. As a result, we combined (1) the log variance of signals, (2) the AR coefficients, (3) the Shannon entropy on the terminal nodes of maximal overlap, and (4) multiscale variance of maximal overlap as a feature vector to prevent bias caused by lost information and to extract sufficient characteristics of ENF signals. The details of the feature vector are shown in Table 3.

5.2. Signal Classification on Extracted Features. In order to find the power grid based on the features extracted from the section, we experimented two different types of classifiers. One is support vector machine-based approach, and the other is XGBoost based classifier.

5.2.1. Support Vector Machines (SVMs). In order to classify the power grids, we applied support vector machines (SVMs) to the feature extracted from Section 5.1. The SVM is a supervised learning model for classification, regression, and novelty detection. One of the SVM's notable characteristics is that the model parameters are determined by a convex optimization, so any local solution is also a global optimum. Given labeled training data, the SVM algorithm finds the maximal margin hyperplane which determines the classification of the new examples. Before training the classifier, we standardized the feature vectors which described in Table 3 and then put into as SVM input vectors. Basically, the SVM is a 2 class classifier. However, we need to classify 3 classes (Eastern, Western, and Texas power grids) more than 2 classes. The values of the regularization item in SVM is set to 1. Accordingly, we employed the one versus all multiclass SVM which considers all possible class pairs to learn. Furthermore, to effectively perform nonlinear classification, the polynomial kernel function with order q was applied as $\text{kern}(x, y) = (1 + x^{\tau} y)^q$, where x and y are the p dimensional vectors. The polynomial kernel of order 2 was chosen in our experiment.

Table 3: Feature vector components for ENF segment.

Index	Extracted features
1	log(variance) of ENF segment
2 ~ 13	AR(12) model parameter
14 ~ 17	Shannon entropy on the MODWPT at level 2
18 ~ $(18 + [\log_2 N + 1])$	log(multiscale variance) of MODWT

N, the number of data samples.

5.2.2. XGBoost. We experimented with another classifier besides SVM to classify the ENF signals. We employed XGBoost [13] which is a scalable end-to-end tree boosting system. The XGBoost, which is an open-source library, is an optimized distributed gradient boosting library. Moreover, the XGBoost proposes a parallel tree learning. The XGBoost provides various parameters for training the model. In this paper, we set the maximum tree depth for base learners as 3,

the value of gamma and alpha as 0, the lambda as 1, and boosting learning rate as 0.1.

6. Implementation and Evaluation

In this section, we describe the results of our proposed approaches with ENF signals which are obtained from power line and multimedia. For both type of ENF signal datasets, we measured the accuracy and F1 score of the power grid estimation according to the number of data points. Moreover, we compared the performance of two types of classifiers: support vector machines and XGBoost.

6.1. Software and Hardware Specifications. All of our experiments were implemented using two PCs. One of two PCs is equipped with an Intel Core i7-7700 CPU (3.60 GHz) with 8 cores, 16 GB memory, and Ubuntu 16.04 LTS (64 bit). The other PC consists of an Intel(R) Core(TM) i5-3230M CPU (2.60 GHz) with 2 cores, 8 GB (1600 MHz DDR3) memory, 250 GB SATA hard drive, and macOS 10.14. We used Python to gather ENF data and to implement "XGBoost" [13]. Additionally, we applied MATLAB for data preprocessing, feature extraction, and SVM. The FFmpeg [27] was employed to extract audio data from videos.

6.2. Evaluation of ENF Signals Extracted from Distribution Level. In Section 5.1, we suggested a feature extraction algorithm for ENF signals. The purpose of this research is not just to extract the features but to classify them into the appropriate power grid: the Eastern, the Western, or the Texas Interconnection. In order to train the model and test the unlabeled data using the introduced classifiers, support vector machines and XGBoost, we randomly divided the datasets into test and train datasets to evaluate the proposed algorithms. When dividing the datasets into training and test datasets, we separated them equally by class so that they maintain the same ratio as the total dataset. We used ENF signal extracted from the distribution level which is described in Section 4.1 as dataset.

We trained 70% and tested 30% of our dataset. For the accurate validity of the results, we repeated these experiments for 20 rounds. In Table 4, the number of data points means the length of data points of the raw signal before extracting the feature. We examined the accuracy conducting on the ENF signal length. As can be seen in Table 4, the longer the signal is used as an input data, the more accurate it is in our proposed method. Furthermore, if the time of the raw ENF signals is longer than one hour, we can accurately predict the power grid more than 88.39%. In spite of accuracy, our proposed method with SVM classifier is little bit higher than the XGBoost classifier in all analyzed time lengths. Based on the F1 score in Table 5, SVM is not always better than XGBoost. However, the F1 score difference between SVM and XGBoost is almost similar to 0.01 or less (Table 5).

6.3. Evaluation of ENF Signals Extracted from Online Multimedia. In order to evaluate the location estimation in multimedia, the multimedia data which have the ENF signals were selected to configure the dataset (Section 4.2). The total number of video files used in the dataset was 374. When dividing the train dataset and the test dataset, we took into account the ratio of each class in the entire dataset. In case of multimedia files, the ENF signals are obtained by using QIFFT and multitone harmonic methods. Afterwards, a feature vector was constructed using the proposed method in Section 5.1. In order to evaluate our suggested feature extraction algorithm and two different classifiers, SVM and XGBoost, we calculate the accuracy of location estimation based on the number of data points used in multimedia (Figure 5 and Algorithm 1).

When calculating the accuracy according to the experimented data points, the accuracy was measured up to 15 minutes, 30 minutes, and an hour unit, because there were less than three video files recorded over 2 hours (Table 2). Each row of Table 6 represents the accuracy and F1 score of $S_j = 60$ for 374 multimedia files, $S_j = 120$ for 199 multimedia files, and $S_j = 240$ for 20 multimedia files. In Table 6, the performance of XGBoost classifier is better than SVM in the entire length of multimedia files. In Table 4, both classifiers display that the longer the length of the ENF signal analyzed from the data collected from the power line level, the more stable performance was obtained. However, when the power grid is estimated using SVM in the multimedia dataset, the accuracy and the F1 score of the multimedia files, which are recorded over 1 hour, were suddenly decreased to 68.82%. Since the XGBoost has increased accuracy and F1 score for the same data, XGBoost is more efficient than SVM in classifying ENF signals.

We analyzed the relationship between the number of harmonic frequencies used to extract ENF signals in multimedia and their accuracy. Moreover, the relationship between the accuracy and frequency band used for the restoration of the ENF signal is analyzed. When we obtained ENF signals from multimedia, we used the multitone harmonics method. Therefore, some multimedia files extracted ENF signals in both fundamental frequency and harmonic frequency bands, totally 8 harmonic bands are employed to extract ENF signals. However, the ENF signals could be extracted from the fundamental frequency and some harmonic frequency bands, only the fundamental frequency or only from the harmonic frequency bands. As shown in Table 1, we extract the ENF signals from multimedia files using 2 to 8 harmonics frequency bands.

The result of the accuracy using XGboost is shown according to the number of harmonic frequency bands used for ENF signal extraction in Table 7. As can be seen in Table 7, the accuracy is not low and not high according to the number of harmonic bands used for ENF signal. Comprehensively considering Table 7 and Figure 2, the accuracy does not have a meaningful relationship depending on whether signal extraction of a specific frequency band is applied or not. Therefore, it is difficult to conclude that the number of frequency bands extracted from multimedia files or participation of a specific frequency band affects the accuracy of experiment.

The accuracy of experiment is more affected by whether data loss occurs when collecting multimedia data,

Table 4: The accuracy of power grid estimation on power line signal (trained 70%, averaged over 20 rounds).

Time (hour)	0.25	0.5	1	2	3	4	6	8	12	24
Number of data points	60	120	240	480	720	960	1440	1920	2880	5760
Proposed with SVM (%)	**67.14**	**76.74**	**88.39**	**95.21**	**96.47**	**98.07**	**99.01**	**99.07**	**99.44**	**99.78**
Proposed with XGBoost (%)	66.36	78	87.67	94.42	95.88	97.06	98.28	98.61	99.55	98.89

Table 5: The F1 Score of power grid estimation on power line signal (trained 70%, averaged over 20 rounds).

Time (hour)	0.25	0.5	1	2	3	4	6	8	12	24
Number of data points	60	120	240	480	720	960	1440	1920	2880	5760
Proposed with SVM	**0.66**	0.76	**0.88**	**0.95**	**0.96**	**0.98**	**0.99**	**0.99**	**0.99**	**0.99**
Proposed with XGBoost	**0.66**	**0.77**	0.87	0.94	0.95	0.97	0.98	**0.99**	**0.99**	0.98

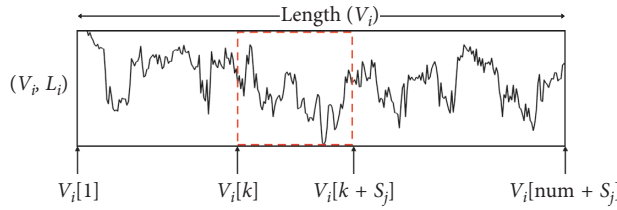


Figure 5: Notation for ENF signals extracted from multimedia where length$(V_i) \geq S_j$. The ENF signals obtained from one multimedia is represented by V_i. The label of the area where multimedia is recorded is indicated by L_i. S_j $(j = 1, 2, \ldots, J)$ is the number of data samples to estimate the location. The *num* is equal to length$(V_i) - S_j + 1$.

INPUT:
$V = \{V_i\}$ consists of ENF signals from videos $(i = 1, 2, \ldots, I)$
$L = \{L_i\}$ is the real label of videos $(i = 1, 2, \ldots, I)$
$S = \{S_j\}$ is the number of data points to analyze $(j = 1, 2, \ldots, J)$

OUTPUT:
$A = \{A_j\}$ is the location estimation accuracy according to analyzed data points $(j = 1, 2, \ldots, J)$

```
(1)  for all S_j ∈ S do
(2)    cnt_num ⟵ 0
(3)    cnt_correct ⟵ 0
(4)    cnt_video ⟵ 0
(5)    for all V_i ∈ V do
(6)      if length(V_i) < S_j then
(7)        break
(8)      end if
(9)      num ⟵ length(V_i) − S_j + 1
(10)     cnt_num ⟵ cnt_num + num
(11)     cnt_vi deo ⟵ cnt_video + 1
(12)     for k = 1 to num do
(13)       predLabel ⟵ predict(v_i[k : k + S_j])
(14)       if predL abel == L_i then
(15)         cnt_correct ⟵ cnt_correct + 1
(16)       end if
(17)     end for
(18)   end for
(19)   A_j ⟵ 100 × (cnt_correct/cnt_num)
(20) end for
```

Algorithm 1: Calculate accuracy from multimedia.

rather than how many frequency bands can be extracted from the multimedia data. Most of the multimedia data we collected are live webcam service's data. Therefore, those video files were uploaded to the server in real time and we downloaded the files. Considering these characteristics of the video files, data loss could have occurred due to delays caused by the network or external environment. If data loss occurs when downloading a file, our proposed method which analyzes the time and frequency domains is hard to predict the correct result. Consequently, in order

Table 6: The accuracy and F1 score of power grid estimation on multimedia data (trained 70%, averaged over 20 rounds).

Time (hour)	Number of data points (S_j)	Number of multimedia	Accuracy		F1 score	
			Proposed with SVM (%)	Proposed with XGBoost (%)	Proposed with SVM	Proposed with XGBoost
0.25 ~	60	374	83.23	**85**	0.75	**0.79**
0.5 ~	120	199	82.92	**94.17**	0.75	**0.93**
1 ~	240	20	68.82	**99.92**	0.56	**0.99**

Table 7: The accuracy according to the number of harmonic bands extracted from multimedia files with XGBoost (trained 70%, averaged over 20 rounds).

Time (hour)	Accuracy by the number of harmonic bands (%)								Average
	1	2	3	4	5	6	7	8	
0.25 ~	—	**99.07**	**98.91**	**99.25**	49.25	59.86	61.84	**88.21**	85
0.5 ~	—	**99.74**	**99.95**	**100**	73.85	85.63	87.79	**95.22**	94.17
1 ~	—	**99.88**	—	—	**100**	**100**	99.68	**99.97**	99.92

to estimate a location using multimedia files, the data loss should be small and the file should not be modified.

7. Discussions

In this paper, we have studied about estimating the location using feature extraction and classification methods from ENF signals. The ENF signals used in this paper are extracted from the power system distribution level and online multimedia. In this section, we describe our research can be applied to discussion of location privacy issues and digital forensics. Furthermore, we discuss the ways to reduce the influence of ENF signals when recording videos or audios. We also explain about the limitations and future work of our research.

7.1. Location Privacy. The location information is one of the most important personal information. Therefore, most countries recommend that the web service provider should have user's consent to gather user's location information. We showed that we could find out the power grid of recorded video files which are uploaded in three worldwide live webcam services, such as Skyline Webcams [9], Earthcam [8], and explore [10] and one global video sharing web service YouTube [7]. If the video files or audio files are taken in a good environment such as using low noise environment and high quality equipment, the location can be found with a higher probability. Even if the user does not agree with the location information when uploading video files or audio files, the location of the recorded files can be estimated by our proposed method. This is a matter of caution as video sharing services and personal broadcasts using the Internet are increasing worldwide.

7.2. Digital Forensics. Since the ENF signals have unique properties in location and time, it can be used for digital forensics. In particular, the applicability of ENF signals has been continuously raised in determining the authenticity of digital audio recordings (Cooper [28], Grigoras [3], and Grigoras [4]). However, in those studies, a reference ENF database which consists of ENF signals extracted in the same area and the same time as the digital audio recording was required to determine the authenticity. Since the authenticity of a file is generally progressed after recording, it is difficult to determine the modification of the file if there are no reference data. This means that, in order to actually apply these researches to digital forensics, reference data must always be constructed. This is not a simple matter in terms of time and economy. However, even if there are no ENF signals extracted at the same time of recording an audio file, we can determine the modification of multimedia files by applying our algorithm. By using the feature extraction techniques and the multiclassification technique proposed in this paper, the ENF signals can be more efficiently applied to the digital forensics.

7.3. Countermeasure. In terms of information security, the fact that the ENF signal is stored when recording video files or audio files can cause a problem of privacy leakage of users. The quality of the ENF signals stored in the files is highly affected by the surroundings and equipment at the time of recording. Hajj-Ahmad et al. [29], Fechner and Kirchner [30], Brixen [31], and Chai et al. [32] studied about the factors that capture the ENF signals in audio files or video files. They could capture the ENF signals when recording with dynamic microphones, but not with electret microphones because of the electromagnetic fields. In addition, they stated that the factors that capture ENF signals are related to the internal compression of the recorder. If the recorder internal compression is strong, it can reduce the ENF signals in videos or audios. Furthermore, if the recorded video files or audio files are modified such as deleting or inserting fake signals, it will be hard to obtain the information.

7.4. Limitations and Future Work. We collected data and experimented only on power grids in the United States. Since there are only three power grids in the United States, it is

necessary to carry out our experiments on more power grids in order to confirm our proposed methods. In addition, we need to supplement the technique of classifying the areas in the same power grid because wide areas are made up of a single power grid. Although the ENF signals have originality in terms of time and location, many related studies about classifying ENF signals are limited to classifying locations, including our studies. Therefore, we will also study on techniques for extracting features and classifying ENF signals over time. If the features of the ENF signals can be extracted by time, it is possible to infer both the time and region where an arbitrary signal is extracted. This will further enhance the value of the ENF signals as digital forensics.

In future work, we will collect more video files and audio files from the Internet and examine our proposed algorithms. In this paper, we examined 374 video files collected from web services. In order to analyze the accuracy according to the harmonic frequency bands, the number of videos was insufficient and it was difficult to generalize. Therefore, we will gather more video files and audio files from the Internet and then improve our algorithms. Furthermore, the research about divide and conquer technique can help to detect the modification of the multimedia file and improve our algorithm. The divide and conquer method is an approach to tough problems divide into several problems. It is widely used to solve the classification or forecasting problems [33–35].

8. Conclusion

In this paper, we propose an algorithm to identify the power grid using the ENF signals extracted from the power system's distribution level and online multimedia such as video files and audio recordings. In order to find out the power grid from ENF signals, we extracted the feature from the ENF signal using three main feature extraction algorithms and found out the power grid using the classifier. We used two types of datasets to evaluate the validity of our proposed algorithm. The first dataset is ENF signals extracted using the FDR equipment at the distribution level of the power system. Another dataset is the ENF signal from the video files and audio files uploaded on the Internet. For each dataset, the accuracy of the proposed algorithm was measured according to various time units such as 15 minutes, 30 minutes, and an hour. In case of the ENF signals extracted from the power line and multimedia files, the more the number of analyzed samples, the higher the accuracy of estimation results when employing XGBoost classifier. Consequently, this paper is notable in that it can estimate the power grid from video files and audio files uploaded on the Internet as well as the ENF signals collected from the power line.

Data Availability

All data used in the paper come from cited references or are reported in the paper

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (no. R7117-16-0161; Anomaly Detection Framework for Autonomous Vehicles).

References

[1] B. Liscouski and W. Elliot, "Final report on the August 14, 2003 blackout in the United States and Canada: causes and recommendations," vol. 40, US-Canada Power System Outage Task Force, Washington, DC, USA, 2004, Tech. Rep. 4, https://emp.lbl.gov/publications/final-report-august-14-2003-blackout.

[2] A. Hajj-Ahmad, R. Garg, and M. Wu, "ENF based location classification of sensor recordings," in *Proceedings of the 2013 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 138–143, IEEE, Guangzhou, China, November 2013.

[3] C. Grigoras, "Digital audio recording analysis: the electric network frequency (ENF) criterion," *International Journal of Speech, Language and the Law*, vol. 12, no. 1, pp. 63–76, 2005.

[4] C. Grigoras, "Applications of ENF criterion in forensic audio, video, computer and telecommunication analysis," *Forensic Science International*, vol. 167, no. 2-3, pp. 136–145, 2007.

[5] Facebook, "Facebook," March 2019, https://www.facebook.com.

[6] Instagram, "Instagram," March 2019, https://www.instagram.com.

[7] YouTube, "YouTube," March 2019, https://www.youtube.com.

[8] EarthCam, "EarthCam," March 2019, https://www.earthcam.com.

[9] Skyline Webcams, "Skyline Webcams," March 2019, https://www.skylinewebcams.com.

[10] Explore, "Explore," March 2019, https://explore.org.

[11] Y. Jeon, M. Kim, H. Kim, H. Kim, J. H. Huh, and J. W. Yoon, "I'm listening to your location! inferring user location with acoustic side channels," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pp. 339–348, International World Wide Web Conferences Steering Committee, Lyon, France, April 2018.

[12] H. Kim, Y. Jeon, and J. W. Yoon, "Construction of a national scale ENF map using online multimedia data," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 19–28, ACM, Singapore, November 2017.

[13] T. Chen and C. Guestrin, "XGBoost: a scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, ACM, San Francisco, CA, USA, August 2016.

[14] A. Hajj-Ahmad, R. Garg, and M. Wu, "ENF-based region-of-recording identification for media signals," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1125–1136, 2015.

[15] W. Yao, J. Zhao, M. J. Till et al., "Source location identification of distribution-level electric network frequency signals at multiple geographic scales," *IEEE Access*, vol. 5, pp. 11166–11175, 2017.

[16] Ž. Šarić, A. Žunić, T. Zrnić, M. Knežević, D. Despotović, and T. Delić, "Improving location of recording classification using electric network frequency (ENF) analysis," in *Proceedings of the 2016 IEEE 14th International Symposium on Intelligent Systems and Informatics (SISY)*, pp. 51–56, IEEE, Subotica, Serbia, August 2016.

[17] A. Hajj-Ahmad, R. Garg, and M. Wu, "Instantaneous frequency estimation and localization for ENF signals," in *Proceedings of the 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*, pp. 1–10, IEEE, Hollywood, CA, USA, December 2012.

[18] M. Abe and J. O. Smith III, "Design criteria for simple sinusoidal parameter estimation based on quadratic interpolation of FFT magnitude peaks," in *Audio Engineering Society Convention 117*, Audio Engineering Society, New York, NY, USA, 2004.

[19] G. Hua, G. Bi, and V. L. L. Thing, "On practical issues of electric network frequency based audio forensics," *IEEE Access*, vol. 5, pp. 20640–20651, 2017.

[20] Y. Liu, S. You, W. Yao et al., "A distribution level wide area monitoring system for the electric power grid-FNET/GridEye," *IEEE Access*, vol. 5, pp. 2329–2338, 2017.

[21] Y. Zhang, P. Markham, T. Xia et al., "Wide-area frequency monitoring network (FNET) architecture and applications," *IEEE Transactions on Smart Grid*, vol. 1, no. 2, pp. 159–167, 2010.

[22] Power Information Technology Laboratory at the University of Tennessee, "FNET/grideye web display," 2018, http://fnetpublic.utk.edu/index.html.

[23] E. A. Maharaj and A. M. Alonso, "Discriminant analysis of multivariate time series: application to diagnosis based on ECG signals," *Computational Statistics & Data Analysis*, vol. 70, pp. 67–87, 2014.

[24] D. B. Percival and H. O. Mofjeld, "Analysis of subtidal coastal sea level fluctuations using wavelets," *Journal of the American Statistical Association*, vol. 92, no. 439, pp. 868–880, 1997.

[25] T. Li and M. Zhou, "ECG classification using wavelet packet entropy and random forests," *Entropy*, vol. 18, no. 8, p. 285, 2016.

[26] A. T. Walden and A. C. Cristan, "The phase–corrected undecimated discrete wavelet packet transform and its application to interpreting the timing of events," *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1976, pp. 2243–2266, 1998.

[27] FFmpeg Developers, "FFmpeg," March 2019, https://ffmpeg.org.

[28] A. J. Cooper, "The electric network frequency (ENF) as an aid to authenticating forensic digital audio recordings–an automated approach," in *Proceedings of the 33rd International Conference: Audio Forensics-Theory and Practice Audio Engineering Society Conference*, Audio Engineering Society, Denver, CO, USA, June 2008.

[29] A. Hajj-Ahmad, C.-W. Wong, S. Gambino, Q. Zhu, M. Yu, and M. Wu, "Factors affecting ENF capture in audio," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 2, pp. 277–288, 2019.

[30] N. Fechner and M. Kirchner, "The humming hum: background noise as a carrier of ENF artifacts in mobile device audio recordings," in *Proceedings of the 2014 Eighth International Conference on IT Security Incident Management & IT Forensics*, pp. 3–13, IEEE, Münster, Germany, May 2014.

[31] E. B. Brixen, "Techniques for the authentication of digital audio recordings," in *Audio Engineering Society Convention 122*, Audio Engineering Society, New York, NY, USA, 2007.

[32] J. Chai, F. Liu, Z. Yuan, R. W. Conners, and Y. Liu, "Source of ENF in battery-powered digital recordings," in *Audio Engineering Society Convention 135*, Audio Engineering Society, New York, NY, USA, 2013.

[33] Y. Zhou, T. Li, J. Shi, and Z. Qian, "A CEEMDAN and XGBOOST-based approach to forecast crude oil prices," *Complexity*, vol. 2019, Article ID 4392785, 15 pages, 2019.

[34] Y.-X. Wu, Q.-B. Wu, and J.-Q. Zhu, "Improved EEMD-based crude oil price forecasting using LSTM networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 516, pp. 114–124, 2019.

[35] J. Kevric and A. Subasi, "Comparison of signal decomposition methods in classification of EEG signals for motor-imagery BCI system," *Biomedical Signal Processing and Control*, vol. 31, pp. 398–406, 2017.

Hindawi
Security and Communication Networks
Volume 2019, Article ID 8565397, 12 pages
https://doi.org/10.1155/2019/8565397

Research Article

Evaluating the Impact of Name Resolution Dependence on the DNS

Haiyan Xu, Zhaoxin Zhang, Jianen Yan, and Xin Ma

School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China

Correspondence should be addressed to Jianen Yan; yanjianen_hit@163.com

Received 21 March 2019; Revised 12 July 2019; Accepted 20 August 2019; Published 9 September 2019

Guest Editor: Sebastian Schrittwieser

Copyright © 2019 Haiyan Xu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In the process of resolving domain names to IP addresses, there exist complex dependence relationships between domains and name servers. This paper studies the impact of the resolution dependence on the DNS through constructing a domain name resolution network based on large-scale actual data. The core nodes of the resolution network are mined from different perspectives by means of four methods. Then, both core attacks and random attacks on the network are simulated for further vulnerability analysis. The experimental results show that when the top 1% of the core nodes in the network are attacked, 46.19% of the domain names become unresolved, and the load of the residual network increases by nearly 195%, while only 0.01% of domain names fail to be resolved and the load increases with 18% in the same attack scale of the random mode. For these key nodes, we need to take effective security measures to prevent them from being attacked. The simulation experiment also proves that the resolution network is a scale-free network, which exhibits robustness against random failure and vulnerability against intentional attacks. These findings provide new references for the configuration of the DNS.

1. Introduction

Domain name system (DNS) is one of the most important infrastructures on the Internet. When people receive services of the Internet, they usually connect to the remote host by entering a hostname instead of the IP address that is hard to be remembered by users. This design simplifies users operation but requires a powerful and distributed DNS to provide the service of mapping domain names to IP addresses. The mapping is transparent to the user, and the DNS provides the ability to automatically convert. Therefore, the security and reliability of the DNS are vital to the Internet. If the DNS has problems, the Internet applications based on it may be impossible to provide normal services for users, which may lead to significant economic losses.

As one of the largest distributed systems of the world, the DNS is unmatched in its efficiency and popularity. In order to handle the scale problem, the DNS deploys a large number of name servers organized in a hierarchical structure and distributed throughout the world, as shown in Figure 1. In this hierarchy, there are three types of name servers: root servers, top-level servers, and authoritative name servers. Root servers and top-level servers are managed by professional Internet organizations and academic institutions, so they are more stable and safer than the authoritative name servers that store the name mapping information of their own domain. The management style of the authoritative name servers, relying on the organizations themselves or entrusted to the Internet service providers, is relatively loose, which is a weak link in the DNS. There have been some attacks on the root or top-level domain servers, whose security is also the object of concerns of many researchers [1–3]. However, we may ignore the security of many authoritative servers below the top level. Whether some important authoritative servers will be attacked and a large number of domain names' resolution failures will occur is a question we want to verify from a macroperspective.

The DNS manages domains and regions through authorization and basic rules. The authoritative name servers are generally placed in their own management domain. Most of the administrators deploy more authoritative name servers to increase performance and reliability of name

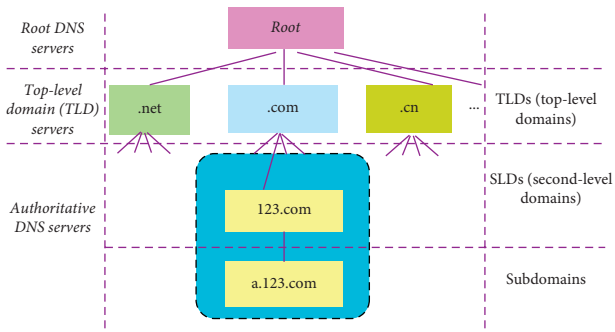


Figure 1: Hierarchical structure of DNS.

resolution. According to the DNS protocol specification [4, 5], servers can be distributed in different domain regions in order to improve the reliability of resolution; however, there is no mechanism to limit the dependencies between domains, which may result in complex name dependence. In this article, we mainly discuss the safety of the authoritative name servers (hereafter referred to as name servers).

We explored the resolving process of 1 million domain names, and the results indicate that 86.14% of the names have interdomain dependence, that is, the resolution of a domain name involves servers located in different domains. Then, in order to study the actual impact of the resolution dependence on the DNS, this paper focuses on the following three aspects:

Firstly, the resolving graph for each domain name is constructed. Thus, for each domain name, especially the one with complex interdomain dependence, its complex relationship resolution dependence is expressed through the graph.

Secondly, the resolving graphs of each domain name are combined to form a global domain name resolution network, which is a complex network of relationships between domain names and name servers. The overall characteristics of the network are also analyzed, reflecting that it is an extremely heterogeneous network, which exhibits robustness against random failures and vulnerabilities against intentional attacks. In addition, the characteristics analysis method of complex network is utilized to mine the key nodes in the network and a metrics of node load is also derived to quantify the node importance in the network.

Thirdly, the impact of critical nodes on the DNS is studied by simulating attacks on the domain name resolution network. Two strategies of the random and core attacks are simulated by node removal method to study how the name dependence impact the DNS, and then the working situation of the whole network is quantified after some name servers have failed. The experimental results show that random attacks on name servers have little impact on DNS as a whole, while attacks on a small number of core nodes in the resolution network have a great impact on DNS.

The paper is organized as follows: Section 2 introduces the concept of domain name resolution dependence, data detection, and the resolving graph of domain names. In Section 3, A global resolution network is constructed and its core nodes mining is introduced. In Section 4, an analysis method is proposed to evaluate the impact of resolution dependence on the DNS. Section 5 gives an overview of the related work in this area. Section 6 concludes our analysis.

2. Domain Name Resolution Dependence

In this section, the definition of domain name resolution dependence, the collection of actual resolution data, the construction of resolving graph for each domain name are introduced.

2.1. Definition of Domain Name Resolution Dependence. *Definition 1.* A domain name u depends on a domain name v, if and only if the resolution results of domain name v impact on the results of domain name *infrastructures*.

The existence of domain name resolution dependence is mainly due to the following three reasons:

(1) Dependent on parent domain: since the resolving process is top-down, a domain name always relies on its parent domain. If without considering the cache, the authoritative data of a domain are always returned by its parent domain. A parent domain may contain more than one subdomain.

(2) Dependent on name servers: the mapping information from hostnames to IP addresses is stored in their name servers. If a resolver wants to resolve a name, DNS queries must be initiated to their name servers.

(3) Dependent on aliases: if a domain name has an alias, the alias must be resolved. Therefore, the resolution of a domain name is also dependent on its alias.

Due to the reasons above, domain dependence is partitioned into the following three types:

(1) Intradomain dependence: if v, an authoritative name server's DNS name of domain u, is administered by domain u itself, then the dependence relationship between u and v is the intradomain dependence, such as the relationship between domain *baidu.com* and its name server *dns.baidu.com*. When the DNS resolver receives this type of resource record, it will use the IP of the name server in the additional part of the resource record to respond for the next query.

(2) Interdomain dependence: if v, an authoritative name server's DNS name of domain u, is not administered by domain u, then the dependence relationship between u and v is interdomain dependence, such as the relationship between the domain *edu.cn* and its name server *cuhk.edu.hk*. When the DNS resolver receives this type of resource record, it will requery the address of the name server and then use the address to make the next query. Even though the additional part of the response packet has an address for the name server, the DNS resolver will ignore it and still requery the address of the name server.

(3) Alias dependence: if a domain name v is the alias of a domain name u, then u is dependent on v, such as the relationship between http://www.baidu.com and www.a.shifen.com. According to the RFC standard [4], the DNS resolver will restart the query for the address of it once the alias record is received.

Because of the existence of interdomain dependence, some name servers may serve for multiple domain names, which can form very complicated dependence relationships among many domain names. This is verified by detecting the resolution data of a large collection of domain names.

2.2. Dataset. The data for this paper are derived from the ranking data of Top 1 million sites from http://www.alexa.com [6], a famous network navigation service provider. These sites are the most popular sites, and they are typical representatives. For each of these domain names, its recursive DNS request messages are constructed and then sent to the DNS servers. From the returned response messages, we recorded the name and address of name servers. If the relationship between a domain name and its name server belongs to the type of interdomain dependence, the recursive detection of its name server will be conducted.

We use an example of a domain name www.edu.cn to explain the method of detecting domain name resolution dependence. Resolving this domain name will be iterated from the root, top-level and *edu.cn* domains [4]. When the domain *edu.cn* is queried, the top-level server returns five authoritative servers, as shown in Figure 2. If it is a server in its own domain, such as *dns.edu.cn*, there are additional records in the DNS response package giving the name server's IP address, then the DNS resolver will issue a DNS query to this address; if the authoritative server is not in its own domain, such as *ns2.cuhk.hk,* there is no IP address of an extraterritorial server. In this case, the server's address needs to be queried iteratively from the root, top-level, and *cuhk.hk* domains. This leads to complex dependencies.

This paper assumes that the root servers and TLD servers were in a normal state, so the resolution dependence is only related to the name servers below the TLD.

In the detection process, this rule is followed: stop detection when the name servers are self-dependent. For example, if the server of *A.net* is *ns1.A.net*, which does not rely on other domains, the detection process will stop.

2.3. Data Statistics for Domain Name Resolution Dependence Measurement. After the resolution dependence detection of 1 million domain names, we find that about 86.14% of the names have interdomain dependence. Then, we further acquire statistics on these data. First, we acquire statistics on the number of name servers involved in each domain name resolution (not including the root name servers and the TLD name servers), and the cumulative distribution of it is plotted in Figure 3. Specifically, about 59.43% of the domain names depend on 2 servers, and

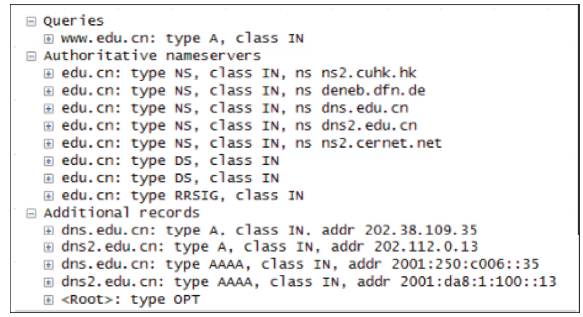


Figure 2: DNS response packets as raw data returned by a top-level server.

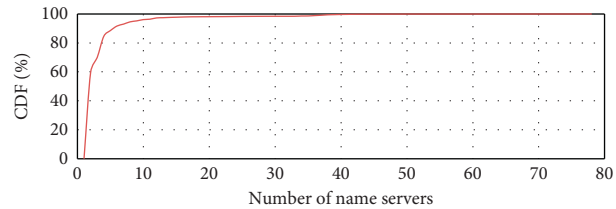


Figure 3: The cumulative distribution of the number of name servers involved in each domain name resolution.

15.38% of them depend on more than 5 servers. The number of servers is in the interval [1, 78]. From the overall distribution, about a quarter of these domains have complex dependencies. The configuration of these domains deserves high attention.

On the other side, some name servers have high degree value. They can provide resolution services for hundreds of domain names, and most of them are the DNS service providers, as shown in Table 1. Once these servers are compromised or down, it may affect a large area of domain names.

As mentioned above, we have demonstrated the overall resolution dependence of the 1 million domain names. We found that some domains have an extremely complex dependency relationship and also discovered that some name servers provide resolution services for hundreds of domain names. These domains and name servers raise our great concerns.

2.4. Construction of the Resolving Graph for Each Domain Name. According to the results of name resolution detection, we construct the resolving graph for each domain name, which reflects the relationship between the domains and their name servers. In this graph, a domain or a name server is considered as a node and the resolution dependence between them is taken as an edge.

Definition 2. Domain name resolving graph can be defined as two tuples:

$$G_d = (V_d, E_d), \tag{1}$$

where V_d is the set of nodes, and any node $v \in V_d$ in the graph represents a domain name or a name server. E_d is the edge

Table 1: Name servers of high dependence degree.

No.	Name servers	Number of domains
1	f1g1ns2.dnspod.net	376
2	f1g1ns1.dnspod.net	376
3	dns3.registrar-servers.com	187
4	dns2.registrar-servers.com	187
5	dns1.registrar-servers.com	187
6	dns4.registrar-servers.com	183
7	dns5.registrar-servers.com	181
8	ns2.bluehost.com	171
9	ns1.bluehost.com	171
10	ns0.dnsmadeeasy.com	170
11	ns1.dnsmadeeasy.com	168
12	ns2.dnsmadeeasy.com	161
13	ns11.dnsmadeeasy.com	161
14	ns3.dnsmadeeasy.com	159
15	ns10.dnsmadeeasy.com	159
16	ns12.dnsmadeeasy.com	158
17	ns2.rackspace.com	152
18	ns13.dnsmadeeasy.com	152
19	ns.rackspace.com	150
20	ns4.dnsmadeeasy.com	148
21	ns2.dreamhost.com	144
22	ns1.dreamhost.com	144
23	dns2.stabletransit.com	139
24	dns1.stabletransit.com	139
25	ns3.dreamhost.com	133
26	ns1.dns.ne.jp	132
27	ns2.dns.ne.jp	130
28	ns14.dnsmadeeasy.com	130
29	dns4.name-services.com	130
30	dns3.name-services.com	130

Number of domains represents the number of domains that depend on this name server to resolve.

set, and any edge $(u, v) \in E_d$ representing that the resolution of domain u depends on the resolution of domain v or relies on a name server.

Figure 4 shows an example of a resolving graph, from which we can see many name servers are involved for mapping a domain name to IP address. The resolution relies on the root server, the TLDs, the *edu*.cn, and its own name servers. It can be viewed as transitive trust behavior from the root servers to the name servers. In this paper, assuming that the root and TLDs are always in normal condition, we only study the impact of authoritative name servers on the DNS. The reason is that the root and the TLDs are managed by professional Internet organizations or academic institutions, while the relatively loose management of the name servers is a weak link. It is clear to see that the resolution of this name "www.edu.cn" relates to the parent domain "*edu.cn*", and it has five name servers. Three of them belong to the type of interdomain dependence. In order to obtain the IP addresses of these three servers, it needs to restart the resolving of the new domains.

When a domain name with interdomain dependence is resolved, many name servers may be involved. Further analysis of the graph can be made to explore the impact of domain name resolution on the existing domain name system at the macrolevel.

3. Name Resolution Network and Its Core Nodes

Since the above resolution dependence data in Section 2.4 is more fragmented and complex, it is not conducive to our macroscopic research on the DNS. Moreover, because of the influence of interdomain dependence, the correlation between different domains is particularly complex. A name server may be used to provide services for different domains. Therefore, we connect the dependence graphs of 400,000 domain names to form a global directed name resolution network, represented as *Net1*. For the sake of clarity, Figure 5 displays an example of a smaller resolution network that contains 100 domain names.

3.1. Characteristics of the Name Resolution Network. After *Net1* is constructed, we calculate some characteristics of it for an overall understanding. Details are shown in Table 2, from which we can see that there are 1,135,448 nodes and 1,811,565 edges in this resolution network. Specifically, a few nodes have a large number of connections, but most nodes have few. This is supported by Figure 6, which is the in-degree distribution of *Net1*. A description of the in-degree is presented in Section 3.2.1. The in-degree distribution reflects that *Net1* is an extremely heterogeneous network, namely, the scale-free network. It exhibits robustness against random failures and vulnerabilities against intentional attacks [7]. In addition, the proportion of the lonely domains (i.e., the domain names only with intradomain dependence) is 15.75% of the 40,000 names, which indicates that the resolving process of three-quarters of them is related to the other domains.

3.2. Identifying the Core Nodes of the Name Resolution Network. We use some classic node centrality measures and a method we proposed to mine the core nodes of the network. The purpose is to identify the important nodes in the network, that is, to discover the name servers providing services for a large number of domain names. Once such a name server becomes the target of DDoS (Distributed Denial of Service), it will inevitably result in the resolution failure of many sites.

3.2.1. Classic Node Centrality Measures. Node centrality measures are a classic tool in complex network analysis to determine the important nodes, and some of them are considered in this paper, such as in-degree, closeness, and Eigenvector centrality. The following is a brief review of these complex network properties, with more background and details in [8].

The in-degree of a node u in a directed network is the number of other nodes that point to u. The closeness of u reflects the proximity between u and other nodes in the network. If the shortest distance between u and the other nodes in the graph is very small, then we think that the closeness of u is high. This measure is more geometrically consistent with the concept of centrality than in-degree centrality. Because if the average shortest distance between

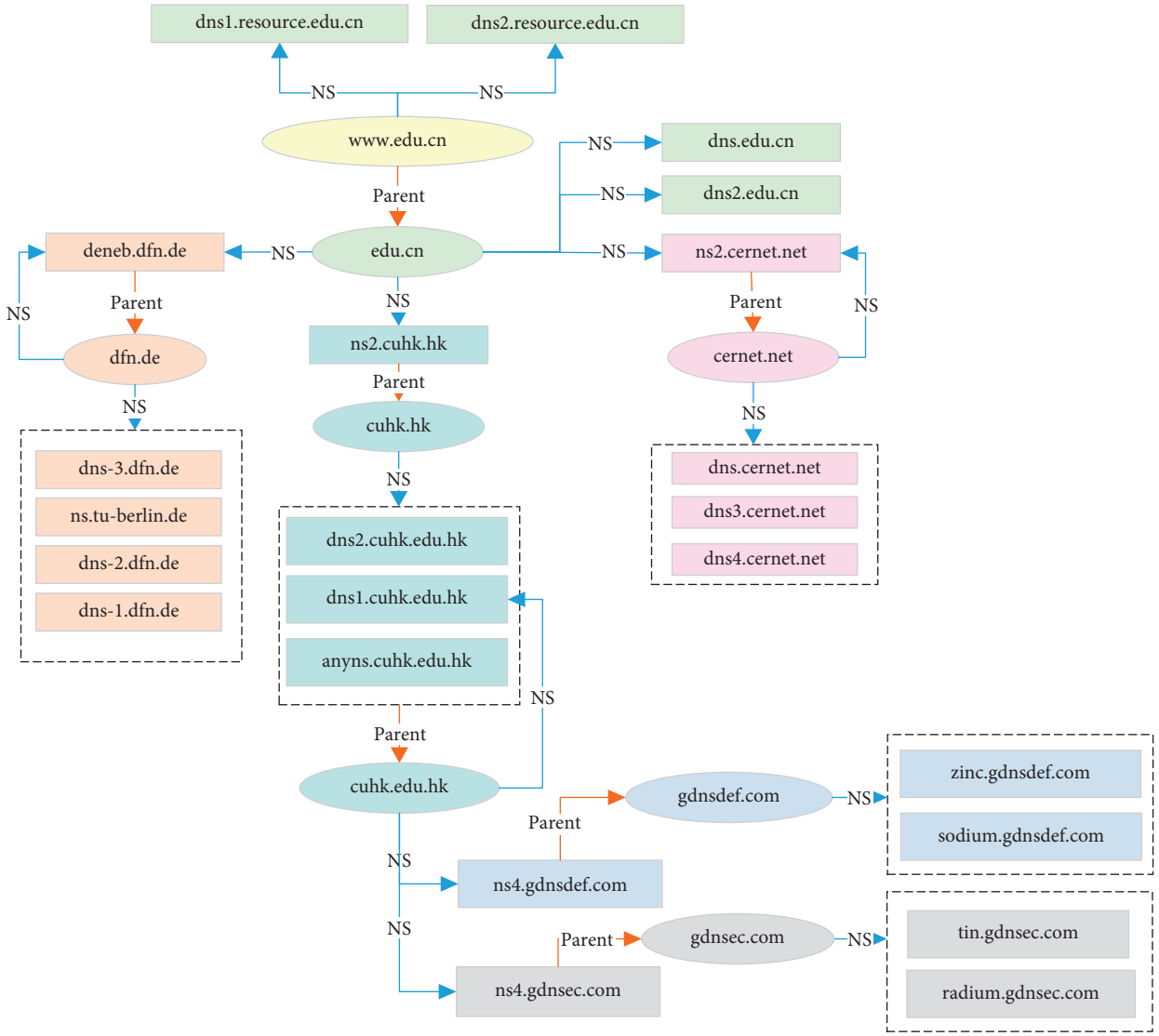


FIGURE 4: Resolving graph of a domain name, where the edge label "parent" indicates it is dependent on the parent domains, the label "NS" indicates it is dependent on the name servers. The resolution of this domain name involves 6 different domains and their name servers, which is symbolized by different colors in the graph. In addition, alias dependence does not appear in this figure. If a domain name has a "cname," alias should be marked on the edge between the domain name and the cname.

node u and other nodes is the smallest, then u is geometrically located at the center of the graph. The Eigenvector centrality of u has a relative index value, which is based on the following principle—contribution of connecting high-scoring nodes to u is more than that of connecting low-scoring nodes. It is the first of the centrality measures that considered the transitive importance of a node in a graph [9].

These classical node centrality measures can identify the important nodes of complex networks from different perspectives. The measures applicable to our resolution network and their specific experimental results are presented in the subsequent analysis in Section 4.4.

3.2.2. Node Load Centrality Measures. Based on the resolution network and combining the actual name resolving process, an algorithm is proposed to quantify the load of each node. For a domain name u, the load of u reflects the sum of the dependence of all other nodes on u in their domain name resolution. The algorithm starts with domain name nodes and traverses each other node in the resolving graph in turn and calculates its value of load. The initial value of the load for all domain name nodes is set as 1 and that for the remaining nodes is set as 0. The calculation of the load is an iterative cumulative process.

As shown in the construction of the resolving graph in Section 2.4, the difference between the types of nodes leads to diverse relations between adjacent nodes, such as relations between (1) child domain and parent domain, (2) domain and name server, and (3) domain name and its alias. Therefore, the quantitative rules for the three categories are defined as follows.

(1) Relations between Child Domain and Parent Domain. As shown in Figure 4, a domain name's resolution is always

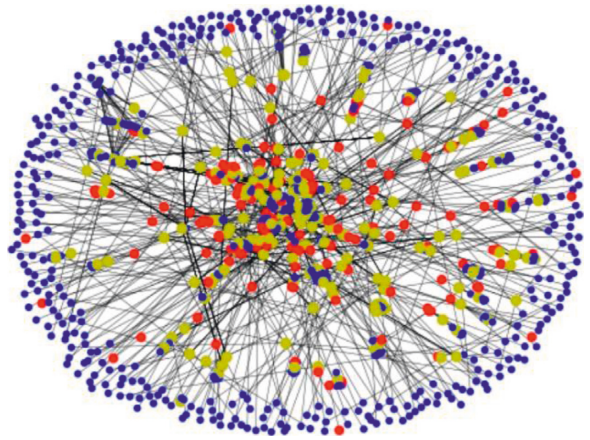


Figure 5: A global dependency graph that contains 100 domain names. The nodes of the domain name and its alias are colored by red, the domain nodes are marked yellow, and the name server nodes are marked blue.

Table 2: Characteristic parameters of *Net1*.

Metric	Value
Number of nodes	1,135,448
Number of edges	1,811,565
Maximum of degree	6,962
Nodes ratio of the degree 1	35.19%
Average value of degree	3.1909
Average value of clustering	0.00006
Proportion of lonely domains	15.75%

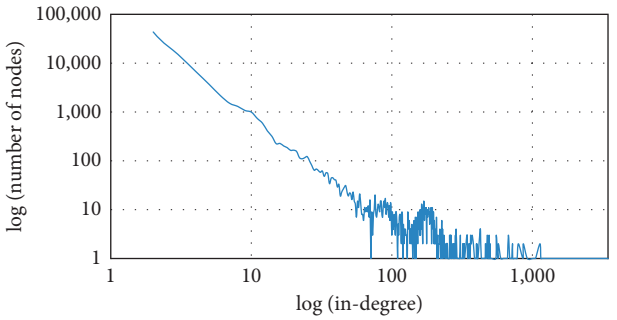


Figure 6: In-degree distribution of *Net1*.

dependent on its parent domain. When a directed edge points from a node to its parent domain node, the load of the parent domain node accumulates the load of its child node. For a parent domain pu, let u_i denote a child domain that depends on pu, and Load_u denote the load of u. Then, the load of parent domain pu in the resolution network is

$$\text{Load}_{\text{pu}} = \sum_{i \in N} \text{Load}_{u_i}, \tag{2}$$

where N is the number of child domains that depend on the parent domain pu.

(2) Relations between Domain and Name Server. A domain name can deploy multiple authoritative name servers. As long as one name server provides normal service, the domain name can be successfully resolved. In this paper, assuming that each name server provides service for its domain with the same probability, the load of a name server node is the load of the domain it is deployed divided by the number of all name servers in that domain. If this name server is set to an authoritative server by more than one domain, we accumulate these values as the load of this name server. Therefore, the load of each name server node in the resolution network is

$$\text{Load}_{\text{ns}} = \sum_{i \in N_{\text{domains}}} \left(\frac{\text{Load}_{\text{un}_i}}{N_i} \right), \tag{3}$$

where ns is a name server node, un_i indicates a domain where *ns* is deployed, N_i is the number of all name servers in domain un_i, and N_{domains} is the number of all domains where ns is deployed.

(3) Relations between Domain Name and its Alias. If a domain name has a resource record of CNAME type, that is, an alias, the alias will inherit its load value. Because at this point the resolution of the domain name has shifted to the resolution of its alias [5]. Generally, an alias corresponds to only one domain name, so there is no cumulative calculation. The load of alias node in the resolution network is

$$\text{Load}_{\text{alias}} = \text{Load}_u, \tag{4}$$

where alias denotes a domain name u's CNAME.

Following the above rules, traverse the nodes of the network and calculate the load for each node. Nodes with larger load are considered to be the center of the network. The results of quantifying the nodes in Figure 4 according to the above method are shown in Figure 7, which only shows a single domain name's dependency relationship, and the nodes in the graph are marked with load values. Since a name server can be used to provide services for different domains, the load of the name server nodes and their parent domain nodes in the resolution network may be accumulated multiple times. Consequently, by comparing the load values, some important nodes in the network can be identified.

Figure 8 presents the top 20 nodes in the network using the centrality measures listed above. The more effective measures in mining the core nodes of the network are discussed in Section 4.4.

4. Evaluating the Impact of Resolution Dependence on the DNS

The global resolution graph is proved to be a scale-free network, which has strong fault tolerance, but its antiattack ability is rather poor for the selective attack based on the key nodes [7]. The presence of the nodes of high-connectivity greatly weakens the robustness of the network. A malicious attacker only needs to select a few nodes of the network to make the network instantly paralyzed. In the DNS, due to the interdomain dependence, the failure of an important node may cause multiple domain names to be unresolved or to

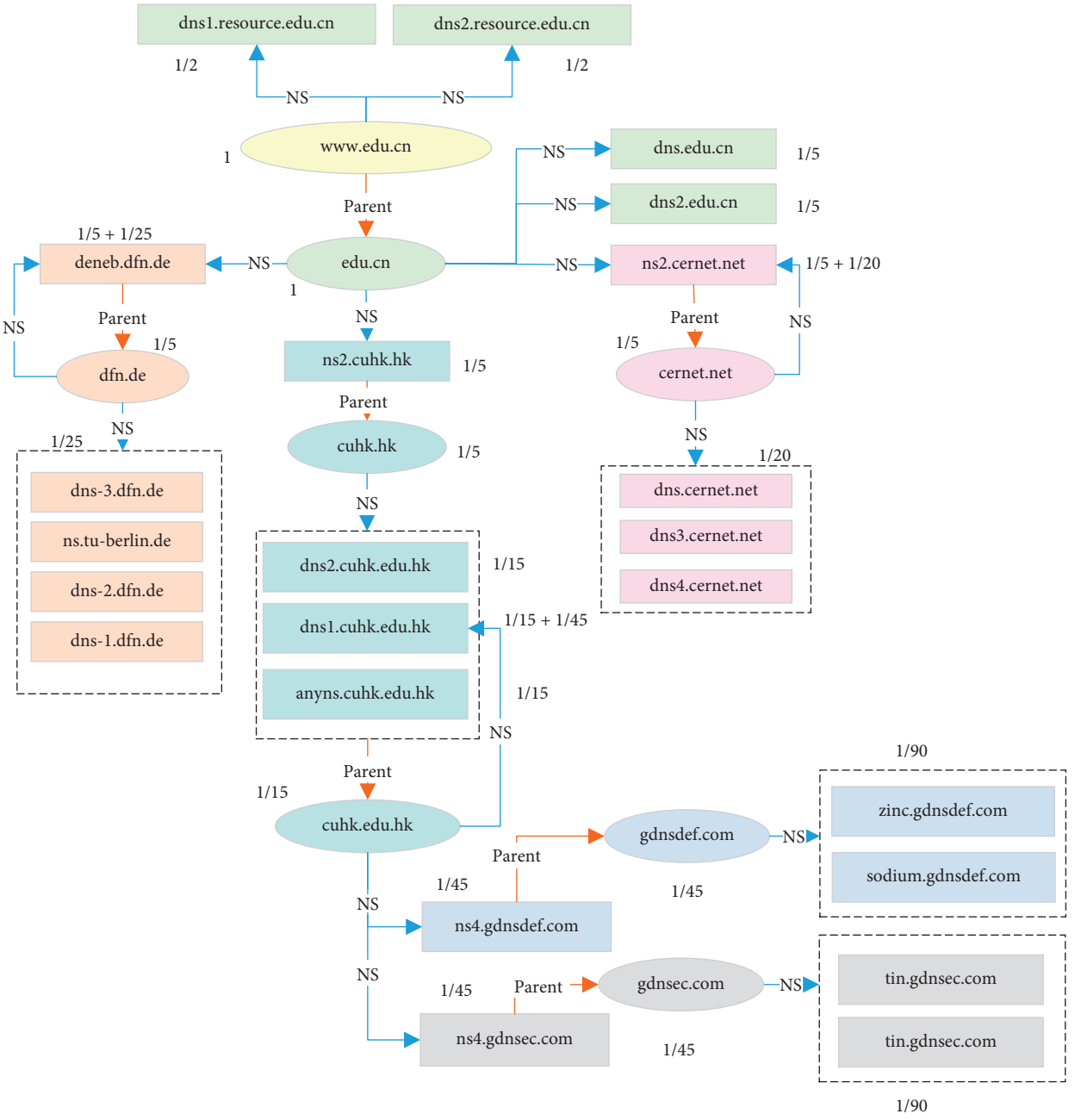


Figure 7: Resolving graph with load value corresponding to Figure 4. The fractional number next to each node represents its load value.

transfer the node's traffic load to its equivalent other servers, which is a cascading failure response in this network. In this paper, we choose to simulate attacks by removing some nodes in the resolution network to evaluate the impact of resolution dependence on the DNS.

4.1. Simulation of Attacks on Name Resolution Networks. On the basis of the name resolution network *Net1* constructed in Section 3, we simulate the DNS attacks by removing some nodes from the network, which is relatively simple compared to simulating the DNS of real resolving process of many domain names. However, it helps us to verify whether the failure of some core name servers will affect the resolution of a large number of domain names at the macrolevel.

In light of different ways of removing nodes, we classify the attacks into two categories: random attacks and core attacks. The random attacks are to select random nodes to remove from *Net1*; the core attacks is to remove the core nodes according to the metrics obtained from the node centrality measures, including the in-degree, closeness, eigenvector, and node load centrality.

After removing the nodes in the network according to certain rules, we evaluate the network status before and after attacks. One of the intuitive and effective indicators used for the evaluation is the name resolution failure rate, which is detailed in Section 4.2. Another indicator is the delay of DNS

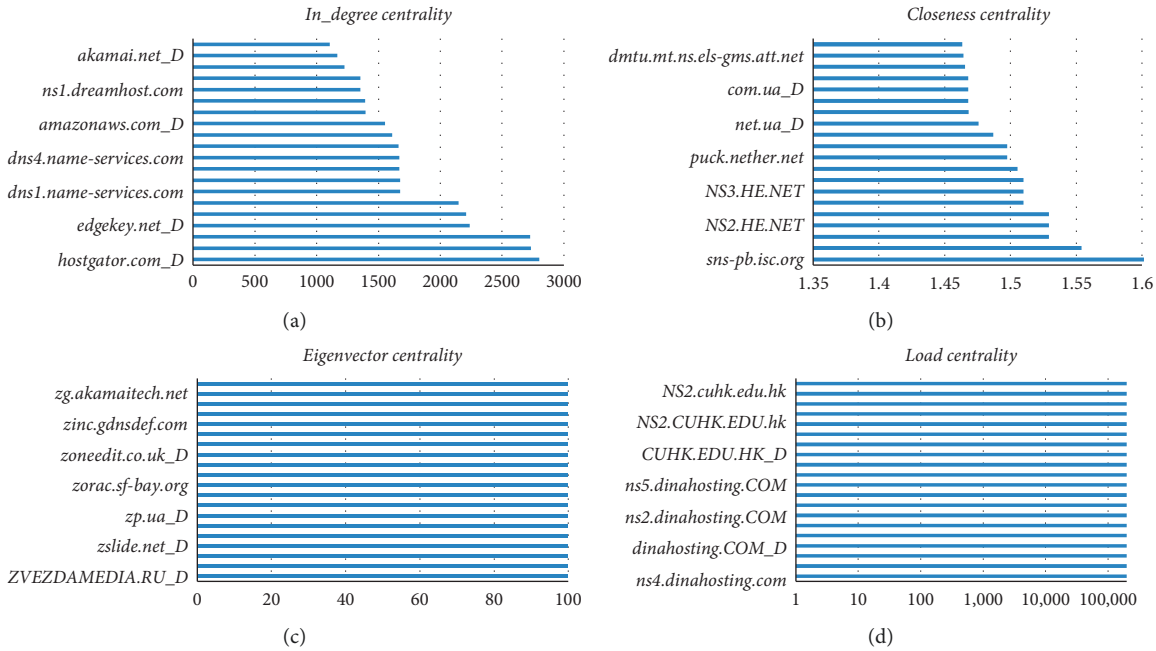


Figure 8: The top 20 nodes of the domain name resolution network, which is mined by measure of in-degree, closeness, eigenvector, and node load centrality.

name resolution, however, which is not adopted in our evaluations. Because *Net1* is a global resolution dependency graph of 400,000 domain names and the actual domain name resolving process does not been emulated, this indicator cannot be obtained from our simulation. In addition, the failure of some server nodes not only causes the related domain name to become unresolvable but also transfers the resolution workload to their equivalent other name servers, which is the cascading failure in this network. Furthermore, we propose an evaluation method of load transfer based on the idea of cascading failure of complex networks, which is described in Section 4.3.

4.2. Resolution Failure Rate Assessment. As the role of the DNS is to provide users with resolution services, the resolution failure rate can be the most intuitive measure of service quality. Consequently, we propose a method to compute the resolution failure rate under attacks from the perspective of static influence. More specifically, after having removed some nodes from the resolution network, we calculate the number of the domain names which become unresolvable and then compute the resolution failure rate. The quantization method is shown in the following formula:

$$E1_{\text{Destruction\_Set}} = \frac{N_{\text{Failure\_Name}}}{N_{\text{All}}}, \tag{5}$$

where Destruction_Set is the set of nodes to be removed from the network, $E1_{\text{Destruction\_Set}}$ is the resolution failure rate caused by the Destruction_Set removed from the network, $N_{\text{Failure\_Name}}$ is the number of domain names that fail to be resolved due to the attack, and N_{All} is the number of all the domain names in the network to be assessed.

4.3. Load Transfer Assessment. In many real-world networks, one or several nodes' failures can cause other nodes to fail through the coupling relationship between nodes, which is called cascading failure. In the name resolution network, there are many combinations of name servers served for resolving a name; that is, the failure of some name servers may not affect the normal resolution of the domain name, but it can aggravate the load of the remaining name servers of the domain name. Accordingly, it is necessary to study the dynamic influence assessment for cascading failure. Here, the changes of load of the overall network before and after attacks are monitored. The calculation of the load has been described in detail in Section 3.2.2. The dynamic impact based on load transfer is as follows:

(1) For each domain name, calculate the average load of all name server nodes that each domain name depends on

$$\text{Load}_{\text{domain}} = \frac{1}{N_{\text{server}}} \sum_{i \in N_{\text{server}}} \text{Load}_{\text{server}_i}, \tag{6}$$

where $\text{Load}_{\text{domain}}$ represents the average load of a domain name, N_{server} represents the number of servers owned by a domain name, and $\text{Load}_{\text{server}_i}$ represents the load of a name server i.

Then, compute the average load for all domain names of the network. The load of the network is shown in the following formula:

$$\text{Load}_{\text{net}} = \frac{1}{N_{\text{domain}}} \sum_{j \in N_{\text{domain}}} \text{Load}_{\text{domain}_j}, \quad (7)$$

where Load_{net} represents the average load of a resolution network, N_{domain} represents the number of domain names in the network, and $\text{Load}_{\text{domian}_j}$ represents the load of a domain name j.

(2) Remove some nodes based on rules, so some domain names become unresolvable and then recalculate the average load in the residual network.

(3) For each attack, compute the change rate of load for the overall network before and after attacks:

$$E2_{\text{Destruction\_Set}} = \frac{\text{Load}_{\text{residual\_net}}}{\text{Load}_{\text{net}}}, \quad (8)$$

where $E2_{\text{Destruction\_Set}}$ is the change rate of load caused by the Destruction_Set removed from the network, Load_{net} represents the load of a resolution network before attacks, and $\text{Load}_{\text{residual\_net}}$ represents the load of a resolution network after attacks.

Thus, we get the specific value of the load change, which is used to represent the degree of cascading effect.

4.4. Experiment and Results. Our experiment is based on the domain name resolution network *Net1* established in Section 3. The DNS attacks are emulated by removing certain nodes from the network. Two strategies of the random and core attacks are simulated to study how the name dependence impacts the DNS. The proportion of nodes in the network being attacked is set to 1%, 5%, 10%, 15%, 20%, etc. The statistical results and the analysis of them are shown below.

4.4.1. Analysis of the Resolution Failure Rate Assessment. The statistical results of the resolution failure rate assessment are displayed in Figure 9, which shows the resolution failure rate of five attacks modes in certain attack proportions. The ordinate represents the resolution failure rate. The abscissa shows the proportion of nodes in the network being attacked. The following conclusions can be drawn from the results:

(1) In the random mode, only 0.01% of domain names fail to be resolved when the attack scale is 1%, and 3.31% fail to be resolved when the scale is 20%. The failure rate is significantly lower than that of other core attacks, indicating that most domain names have multiple resolution paths. Therefore, random small-scale failure does not have a major impact on the DNS.

(2) In the four centrality measures of the core attacks, the effect of in-degree is the best. When the attack scale is 1% by removing the nodes with high in-degree value, 46.19% of the domain names become unresolved. This shows that if a small number of these core nodes are attacked, there will remain a significant impact on the DNS. In addition, the centrality measure of node load proposed in our paper can also better identify the important nodes in the network when the attack scale is above 10%, but in the smaller scale node attacks, it is not as effective as in-degree. Moreover, the closeness and eigenvector centrality are less effective in mining the core nodes of the network than the previous two types.

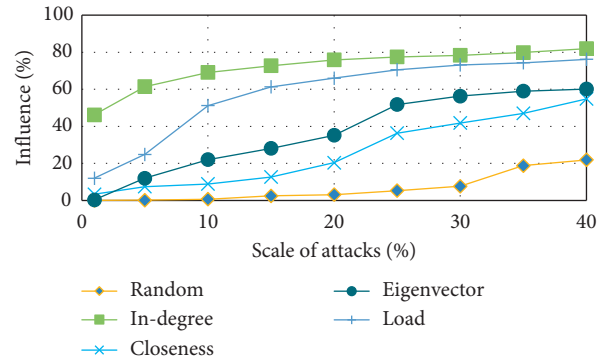


Figure 9: Statistical results of the resolution rate assessment in *Net1*.

4.4.2. Analysis of the Load Transfer Assessment. Figure 10 shows the statistical results of the load transfer assessment. The ordinate indicates the change rate of load after attacks, and the abscissa shows the proportion of nodes in the network being attacked. The following conclusions can be drawn from the analysis of the results:

(1) The results show that when the top 1% nodes (with metric of in-degree centrality) are removed, the load of the network increases by nearly 195%, while the load of the random mode increases by only 18% under the same attack scale. It can be seen that attacks on core nodes will not only cause a large number of domain names to be unresolved but also produce excessive load to other name servers. For these key nodes, effective security measures should be taken to protect them; on the other hand, the workload can be appropriately dispersed to other nodes to avoid the single point failure problems.

(2) In the simulation of core attacks, the effect of in-degree centrality is the best, followed by the load centrality. This is similar to the test results of the resolution failure rate assessment. So, these two measures can be used to identify the core of the resolution network. However, the closeness centrality does not work well in mining core nodes of the name resolution network.

4.4.3. Further Validation. Since all these conclusions need further validation, another set of 200,000 domain names were selected from the 1 million domain names surveyed to form a resolution network, represented as *Net2*. The domain name sets of net1 and net2 are independent, respectively, and have no intersection. The comparison of characteristic

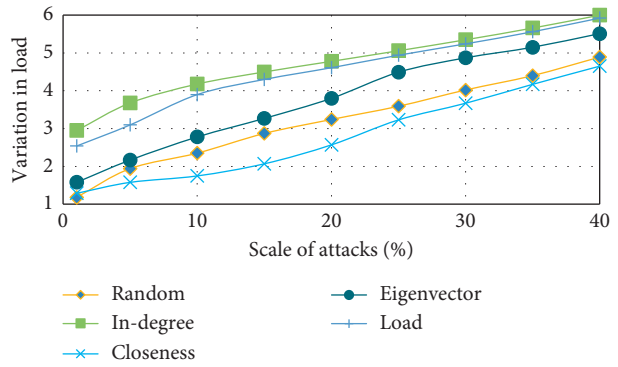


Figure 10: Statistical results of the load transfer assessment in *Net1*.

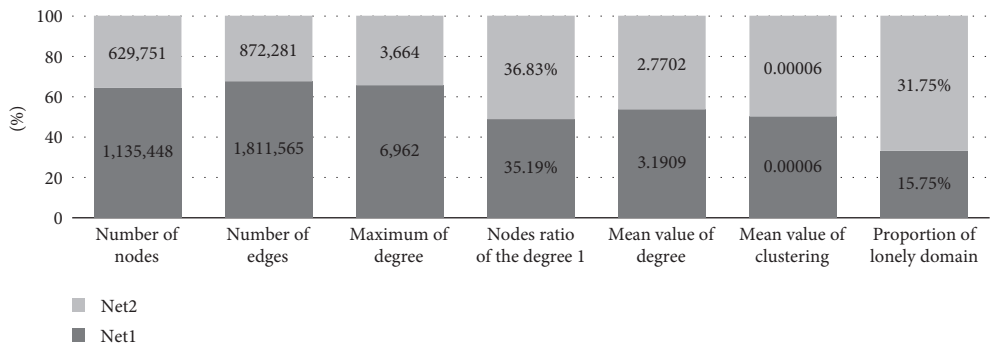


Figure 11: Comparison of characteristic parameters of *Net1* and *Net2*.

parameters of *Net1* and *Net2* is shown in Figure 11, and the in-degree distribution of *Net2* is displayed in Figure 12, showing a few nodes have a large number of connections. This reflects that *Net2* is also a scale-free network, which exhibits robustness against random failures and vulnerabilities against intentional attacks. We use the in-degree centrality and the load centrality measures that present good performance to mine the core nodes, simulate random attacks and core attacks, and utilize the resolution failure rate to evaluate the impact on the network.

The specific results are shown in Figures 13 and 14, from which we can see that the experimental results are consistent with the conclusions of *Net1*. The detailed results of resolution failure rate assessment are as follows: (1) In the random mode, only 0.01% of domain names fail to be resolved when the attack scale is 1% and 4.44% fail to be resolved when the scale is 20%. (2) In the core mode, when the attack scale is 1% by removing the nodes with high in-degree value, 62.93% of the domain names become unresolved and 8 5.38% fail to be resolved when the scale is 20%. This shows that if core nodes are attacked, there will remain a significant impact on the network, but random small-scale failure does not have a significant impact. The results of the load transfer assessment in *Net2* also verified the conclusion.

5. Related Work

The security and availability of the DNS have become a common concern. The current studies in this field mainly

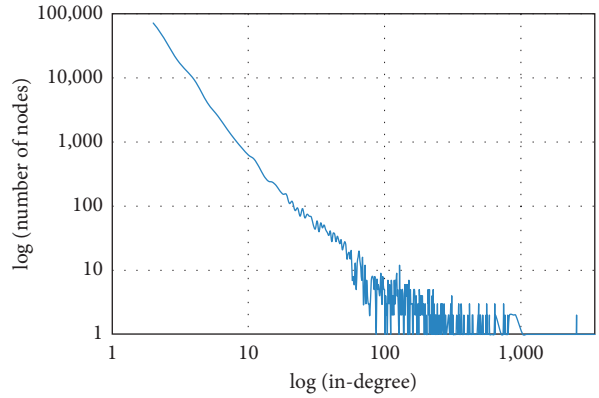


Figure 12: In-degree distribution of *Net2*.

focused on data flow anomaly detection [10–12], DNS amplification attacks [13, 14], DNS servers availability measurement [15, 16], cache poisoning detection [18], DNSSEC security protocols [17–20], and Botnet tracking combined with DNS [21, 22]. Ramasubramanian and Sirer [23] first proposed the concept of DNS dependence, which can lead to a highly insecure naming system. Casey Deccio [24, 25] proposed a DNS dependence model, which featured a probabilistic method to quantify the influence of the domain name in the trusted computing base and used a numerical value from 0 to 1 to quantify the influence degree. Fujiwara et al. [26] took the lead to measure DNS traffic increases due to interdomain dependence, with a result of 60% of DNS traffic involved out-bailiwick name servers,

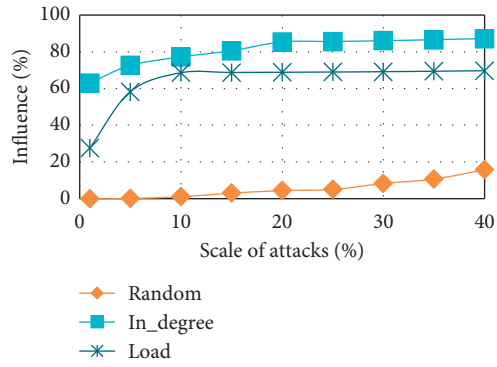


Figure 13: Statistical results of the resolution failure rate assessment in *Net2*.

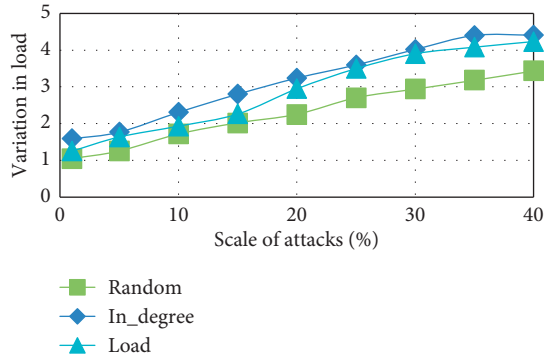


Figure 14: Statistical results of the load transfer assessment in *Net2*.

which increased the resolution delay by about 47% in the actual measurement of the DNS traffic and delay. In summary, DNS interdependent behavior has drawn the attention of researchers in recent years. However, less research has focused on the extent to which name dependence affects DNS.

6. Conclusion

In this paper, we have analyzed the actual impact of the resolution dependence on the DNS, investigated the dependence relationship of 1 million domain names, and found that 86.14% of the domain names are dependent on name servers which are not in their own authorization domain (we do not consider the case of top-level domain and the root domain here). Then, a resolution dependency graph of each domain name has been constructed based on the data we explored. Due to the influence of interdomain dependence, there may be correlations between these graphs. Therefore, based on the graphs of 400,000 domains, a global domain name resolution network has been established to analyze the problem of vulnerability. From an overall perspective, this network is a scale-free network, which exhibits robustness against random failures and vulnerabilities against intentional attacks. This is verified by our simulation of random attacks and core attacks. The resolution failure rate assessment has also been utilized to compute the resolution failure rate, and the load transfer assessment has been employed to calculate the change rate of the load after attacks. The experimental results show that when the key nodes of the first 1% are removed, 46.19% of the domain names become unresolved, and the average load of the residual nodes will increase by nearly 195%, while only 0.01% of domain names fail to be resolved and about 18% of load increase on the same attack scale of the random mode. Moreover, another set of 200,000 domain names were selected from the 1 million domain names to do the same experiment and further evidence the conclusion.

In addition, the classic node centrality measures of the complex network have been introduced and a new method has been proposed to mine the core nodes of the resolution network. In the experiment of simulating the core attack, the effect of these measures is also tested.

These findings provide new references for the configuration of the DNS. DNS administrators should take effective security measures to prevent the core nodes from being attacked. From the macro, it can help to find out weakness and problems in the design of the current domain name system.

Data Availability

The domain name resolution data used to support the findings of this study are currently under embargo, while the research findings are commercialized. Requests for data, 6 months after publication of this article, will be considered by the corresponding author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Science and Technology Major Project under Grant no. 2017YFB0803001 and the National Natural Science Foundation of China under nos. 61370215, 61370211, and 61571144.

References

[1] G. C. M. Moura, R. D. O. Schmidt, J. Heidemann et al., "Anycast vs. DDoS: evaluating the november 2015 root DNS event," in *Proceedings of the 2016 ACM on Internet Measurement Conference*, Santa Monica, CA, USA, November 2016.

[2] Z. Liu, B. Huffaker, M. Fomenkov, N. Brownlee, and K. Claffy, "Two days in the life of the DNS anycast root servers," in *Proceedings of the International Conference on Passive and Active Network Measurement*, Louvain-la-Neuve, Belgium, April 2007.

[3] Y. Xuebiao et al., "DNS measurements at the .CN TLD servers," in *Proceedings of the Sixth International Conference on Fuzzy Systems and Knowledge Discovery*, IEEE, Tianjin, China, August 2009.

[4] P. Mockapetris, Domain Names—Concepts and Facilities, RFC 1034, November 1987.

[5] P. Mockapetris, Domain Names—Implementation and Specification, RFC 1035, November 1987.

[6] Alexa.com[EB/OL], http://www.alexa.com/.

[7] R. Albert, H. Jeong, and A.-L. Barabási, "Error and attack tolerance of complex networks," *Nature*, vol. 406, no. 6794, pp. 378–382, 2000.

[8] L. Lü, D. Chen, X.-L. Ren, Q.-M. Zhang, Y.-C. Zhang, and T. Zhou, "Vital nodes identification in complex networks," *Physics Reports*, vol. 650, pp. 1–63, 2016.

[9] P. Bonacich, "Power and centrality: a family of measures," *American Journal of Sociology*, vol. 92, no. 5, pp. 1170–1182, 1987.

[10] C. Saint-Pierre, F. Cifuentes, and J. Bustos-Jimenez, "Detecting anomalies in DNS protocol traces via passive testing and process mining," in *Proceedings of the IEEE Conference on Communications and Network Security*, pp. 520-521, San Francisco, CA, USA, October 2014.

[11] D. Dagon, "Large-scale DNS data analysis," in *Proceedings of the 2012 ACM conference on Computer and communications security–CCS'12*, pp. 1054-1055, Raleigh, NC, USA, October 2012.

[12] A. Berger, A. D'Alconzo, W. N. Gansterer, and A. Pescapé, "Mining agile DNS traffic using graph analysis for cybercrime detection," *Computer Networks*, vol. 100, pp. 28–44, 2016.

[13] D. C. Macfarland, C. A. Shue, and A. J. Kalafut, "The best bang for the byte: characterizing the potential of DNS amplification attacks," *Computer Networks*, vol. 116, pp. 12–21, 2017.

[14] S. Kim, S. Lee, G. Cho, M. E. Ahmed, J. Jeong, and H. Kim, "Preventing DNS amplification attacks using the history of DNS queries with SDN," in *Proceedings of the European Symposium on Research in Computer Security*, pp. 135–152, Oslo, Norway, September 2017.

[15] E. Casalicchio, M. Caselli, and A. Coletta, "Measuring the global domain name system," *IEEE Network*, vol. 27, no. 1, pp. 25–31, 2013.

[16] H. Gao, V. Yegneswaran, J. Jiang et al., "Reexamining DNS from a global recursive resolver perspective," *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 43–57, 2014.

[17] N. Alexiou, S. Basagiannis, P. Katsaros, T. Dashpande, and S. A. Smolka, "Formal analysis of the kaminsky DNS cache-poisoning attack using probabilistic model checking," in *Proceedings of the 12th International Symposium on High-Assurance Systems Engineering*, pp. 94–103, IEEE, Boca Raton, FL, USA, July 2011.

[18] C. Deccio, J. Sedayao, K. Kant, and P. Mohapatra, "Quantifying and improving DNSSEC availability," in *Proceedings of the 20th International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–7, Maui, Hawaii, July 2011.

[19] R. V. Rijswijk-Deij, A. Sperotto, and A. Pras, "Making the case for elliptic curves in DNSSEC," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 5, pp. 13–19, 2015.

[20] T. Chung, R. van Rijswijk-Deij, B. Chandrasekaran et al., "A longitudinal, end-to-end view of the DNSSEC ecosystem," in *Proceedings of the 26th USENIX Security Symposium*, pp. 1307–1322, Vancouver, BC, Canada, August 2017.

[21] H. Choi and H. Lee, "Identifying botnets by capturing group activities in DNS traffic," *Computer Networks*, vol. 56, no. 1, pp. 20–33, 2012.

[22] J. Kwon, J. Lee, H. Lee, and A. Perrig, "PsyBoG: a scalable botnet detection method for large-scale DNS traffic," *Computer Networks*, vol. 97, pp. 48–73, 2016.

[23] V. Ramasubramanian and E. G. Sirer, "Perils of transitive trust in the domain name system," in *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, pp. 379–384, Berkeley, CA, USA, October 2005.

[24] C. Deccio, J. Sedayao, K. Kant, and P. Mohapatra, "Quantifying DNS namespace influence," *Computer Networks*, vol. 56, no. 2, pp. 780–794, 2012.

[25] C. Deccio, *Quantifying and Improving Dns Availability*, University of California, Davis, CA, USA, 2010.

[26] K. Fujiwara, A. Sato, and K. Yoshida, "DNS traffic analysis: issues of IPv6 and CDN," in *Proceedings of the 12th International Symposium on Applications and the Internet*, pp. 129–137, Izmir, Turkey, July 2012.

Hindawi
Security and Communication Networks
Volume 2019, Article ID 6716918, 8 pages
https://doi.org/10.1155/2019/6716918

Research Article

A Data-Driven Approach to Cyber Risk Assessment

Paolo Santini,[1] Giuseppe Gottardi,[2] Marco Baldi,[1] and Franco Chiaraluce[1]

[1]*Università Politecnica delle Marche, Ancona, Italy*
[2]*Fondazione F3RM1, Milan, Italy*

Correspondence should be addressed to Marco Baldi; m.baldi@univpm.it

Received 5 April 2019; Revised 2 July 2019; Accepted 11 August 2019; Published 9 September 2019

Guest Editor: Hyoungshick Kim

Copyright © 2019 Paolo Santini et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Cyber risk assessment requires defined and objective methodologies; otherwise, its results cannot be considered reliable. The lack of quantitative data can be dangerous: if the assessment is entirely qualitative, subjectivity will loom large in the process. Too much subjectivity in the risk assessment process can weaken the credibility of the assessment results and compromise risk management programs. On the other hand, obtaining a sufficiently large amount of quantitative data allowing reliable extrapolations and previsions is often hard or even unfeasible. In this paper, we propose and study a quantitative methodology to assess a potential annualized economic loss risk of a company. In particular, our approach only relies on aggregated empirical data, which can be obtained from several sources. We also describe how the method can be applied to real companies, in order to customize the initial data and obtain reliable and specific risk assessments.

1. Introduction

The process of risk assessment and treatment is fundamental to the implementation of an effective cyber security program and plays a crucial role for the national and international regulations in the field of data protection. A complete understanding of cyber risks is necessary, in order to ensure that the security controls an organization has in place are sufficient to provide an appropriate level of protection against cyber threats.

However, defining reliable models for the cyber risk exposure is still an open problem. Existing models [1–4] suffer from some important concerns that, for example, prevent the insurability market development [5]. First of all, cyber risk evaluation and the study of its related impact are performed mostly in qualitative ways, which are usually affected by errors and misrepresentations of the risk. They also exhibit several disadvantages, such as the approximate nature of the achieved results and the difficulty of performing a cost-benefits analysis [6]. Quantitative approaches, in their turn, are usually based on scoring systems that associate a certain score to a technological/organizational context. The idea is commendable but, as it will be clarified afterward, the way in which it is commonly implemented does not give a realistic measure of the cyber risk and the related impact. Reliable models for the measure of the cyber risk are not available or have significant limitations, like the lack of generalization and the fact that most works consider only the analysis of past data to derive probabilistic models, while it is not clear how to obtain reliable estimates about future events [7, 8]. Moreover, some quantitative approaches, like the well-known HTMA (how to measure anything) [9] and the FAIR [10] methods rely on a subjective evaluation of the likelihood of an event (in particular, of the probability of a successful attack due to a certain threat) given by a team of experts [11, 12]. These kinds of probabilities usually show some level of inaccuracy and should be replaced by more objective models. The impact of the set of considered threats is then measured in terms of economic loss, which is also subjectively estimated. Based on these premises, the need to improve the quantitative evaluation of the cyber risk of an organization, through a dynamic monitoring of the attacks and vulnerabilities the organization is experiencing, clearly emerges.

A recent study, for the case of data breaches in IT and information security, has been developed in [13]. However, it is clear that the whole panorama involves other sectors (for

instance, business and finance companies) and other cyber threats. This means that in order to obtain reliable representations of this wide scenario, more general analyses are required. A step towards the generalization of this work has been done in [14], where the authors consider the whole range of cyber risks and associated costs, by analyzing sufficiently large and available datasets. This kind of approach is not new in the literature (see, for instance, [8, 15] for a study based on data breaches): through the analysis of a large amount of historical data, probabilistic models about the registered cyber crime events can be derived. However, as the authors in these papers clearly state, such an approach has some important limitations. Indeed, cyber crime is a dynamic and continuously evolving phenomenon: considering only past data is clearly not enough to obtain reliable estimates about future events.

For instance, let us suppose that a company wants to determine its risk exposure and eventually define possible strategies to increase its security against cyber crime. In such a case, relying on probabilistic models is very likely to be not enough, since specific countermeasures and consequent attack strategies should be taken into account. One example of this process is given in [12], where the authors consider a dynamic model that describes possible interactions between a defender and an attacker. In such a model, each countermeasure is considered as a software update of an existing cyber security system and is characterized by an effectiveness score; the optimal strategy is thus obtained by taking into account the interplay between the defender and the attacker.

Previous works highlight the fact that the whole scenario of cyber risk, with the heterogeneous ensemble of all possible players and events, looks like a phenomenon that difficultly allows deriving reliable probabilistic models. In particular, applications on some real case studies might strongly depend on specific aspects of the involved subjects. For instance, conducting a risk assessment with a qualitative approach may result in a good level of controls for malware defense, but this does not provide evidence on whether these security measures are effective in counteracting malware attacks. As a consequence, inaccurate prioritization could result in valuable resources being spent on risk areas that may not be very important and which may not deserve such resources and vice versa.

As mentioned above, one way to reduce the uncertainty in this scenario is the one of relying on opinions of experts [11, 12]. This kind of approach is followed in HTMA [9], where a set of threats is characterized by a likelihood value and the corresponding impact. Basically, the likelihood is the probability of successful attack due to each threat, while the impact expresses the subsequent economic loss. In particular, in the HTMA method, the impacts are estimated through interviews to experts that, for each threat, are asked with the 90% confidence range for possible economic losses. Their answers are then used to define the random variable associated to the impact; a log-normal distribution is assumed for each one of the considered threats.

In this paper, we use the HTMA approach but assuming experts' opinions only as a starting point to be progressively and continuously improved through the acquisition of new and updated information on the organization's behavior against cyber threats. More precisely, we propose to exploit a combination of probabilistic techniques and objective data as an input to HTMA. Our goal is to define a methodology for fitting a probabilistic model into a real case study. The procedure we propose is substantially based on checking the effectiveness of the applied measures through a data-driven approach. As previous studies by SANS Institute already outline [16], one of the key focuses of an effective cyber risk assessment is the measurement of the security controls implementation effectiveness. Indeed, as seen in most risk assessment methods, a risk matrix only representing impact and likelihood is a commonly used tool to assess cyber risk. According to previous approaches, these values are obtained through the analysis of a sufficiently large amount of historical data. However, these values hardly reflect the actual state of a particular entity under analysis. Indeed, a reliable assessment of the risk exposure without taking into consideration the effectiveness of the applied measures is a hard task. For this purpose, our analysis considers the CIS 20 Critical Security Controls (Center for Internet Security, https://www.cisecurity.org/), but the same procedure can be applied to any other suitable set of security issues.

The paper is organized as follows. In Section 2, we briefly remind the probabilistic model we consider, based on the HTMA approach. In Section 3, we describe how empirical data can be used to fit the model into an actual entity (for instance, a company). In Section 4, we show an application of this methodology to some real case scenarios. Finally, in Section 5, we draw some conclusive remarks.

2. Probabilistic Model

In this section, we shortly describe the method we use for evaluating the cyber risk exposure of a company. We first present a generic model that fits into our study case and give the basic notions that are fundamental for our analysis. We then briefly introduce the HTMA methodology and describe how it can be used to derive a quantitative measure for the risk exposure.

2.1. General Model. We consider a set of events $\mathbf{I}_E = \{E_1, \ldots, E_n\}$, the i-th one with probability of occurrence p_i, for $i = 1, \ldots, n$; in this paper, we assume that the events are independent, which means that the occurrence of E_i does not influence the occurrence of all the other events E_j, with $j \neq i$. More complex scenarios, in the presence of correlation between different events, will be analyzed in future work. Each event E_i is also linked to a random variable c_i, which is associated to the impact and is described through a probability distribution $f^{(i)}(c_i)$: each time the event E_i occurs, it has an impact c_i, whose particular value depends on $f^{(i)}(c_i)$. Then, we denote as $C(\mathbf{I}_E)$ the random variable defined as the sum of the impacts of the considered events. Our goal is to characterize the statistical properties of $C(\mathbf{I}_E)$.

Actually, unless specific choices for the distributions of c_i are made, providing a closed form for the distribution of

$C(\mathbf{I}_E)$ is unfeasible. However, we can proceed with numerical simulations and estimate its cumulative distribution function (CDF). More precisely, we can simulate N different scenarios: in each scenario, we simulate the occurrences of the events E_i, and for each occurred event, we randomly extract the corresponding impact, according to $f^{(i)}(c_i)$. Let $C^{(j)}$ be the resulting total impact obtained in the j-th simulated scenario; then, the CDF of $C(\mathbf{I}_E)$ can be estimated as

$$F_{C(\mathbf{I}_E)}(a) = P\{C(\mathbf{I}_E) \leq a\} = \frac{\#_j\{C^{(j)} \leq a\}}{N}. \quad (1)$$

It is clear that, in order to obtain reliable estimates via numerical simulations, the value of N must be sufficiently large.

One crucial quantity in our analysis is the *loss exceedance curve* defined as

$$L_{C(\mathbf{I}_E)}(a) = P\{C(\mathbf{I}_E) \geq a\} = 1 - F_{C(\mathbf{I}_E)}(a) + P\{C(\mathbf{I}_E) = a\}. \quad (2)$$

By definition, the value of $L_{C(\mathbf{I}_E)}(a)$ corresponds to the probability that the total impact of the considered events is equal to or exceeds a threshold value a. If, for instance, we have $L_{C(\mathbf{I}_E)}(a) = 0.2$, then this means that the probability that the set of considered events results in a total impact $\geq a$ is equal to 0.2. Also by definition, we have $L_{C(\mathbf{I}_E)}(0) = 1$ and $L_{C(\mathbf{I}_E)}(\infty) = 0$.

2.2. Cyber Risk Assessment Based on HTMA. In this section, we briefly describe the use of the HTMA method for assessing cyber risk. We consider a set of n different cyber threats and, with reference to the notation introduced in the previous section, define E_i as the event that the i-th threat has resulted in an economic loss for the company. Thus, p_i is the probability of such an occurrence, for instance, in the time interval of one year.

Following the HTMA methodology, each economic impact is obtained on the basis of interviews: in particular, for each threat, experts are asked with the 90% confidence interval of economic losses that their company might sustain, in case of occurrence of the considered threat. The impact of each cyber threat is then associated to a range in the form $[c_i^{(\min)}; c_i^{(\max)}]$, corresponding to the 90% confidence interval. In particular, as mentioned, the HTMA method assumes that each economic impact follows a log-normal distribution, with mean μ_i and standard deviation σ_i that are obtained as

$$\mu_i = \frac{\log\left(c_i^{(\max)}\right) + \log\left(c_i^{(\min)}\right)}{2}, \quad (3)$$

$$\sigma_i = \frac{\log\left(c_i^{(\max)}\right) - \log\left(c_i^{(\min)}\right)}{3.29}, \quad (4)$$

respectively. Thus, starting from likelihood values and the corresponding economic impacts distribution, Monte Carlo simulations can be performed to obtain the loss exceedance curve $L_{C(\mathbf{I}_E)}(a)$.

In each simulated scenario, we consider all possible (i.e., identified) threats. For each threat, we randomly sample a variable with continuous uniform distribution in $[0; 1]$. If such value exceeds p_i, then the corresponding impact is sampled (according to the corresponding log-normal distribution); otherwise, it is set as 0. Then, by computing the sum of all impacts, we obtain the value of the total impact. If we consider a sufficiently large number of scenarios, then we can obtain a reliable estimate of the loss exceedance curve.

3. Cyber Risk Assessment Framework

The loss exceedance curve introduced in the previous section strongly depends on the company assets and characteristics. In other words, in order to obtain a reliable estimate of the loss exceedance curve, we need to customize the values p_i and c_i to the actual organization we are considering. This operation is usually performed through surveys submitted to the company, in order to obtain a reliable overview of what the company is currently doing to prevent cyber threats. However, it is clear that the use of such answers is likely not enough to obtain a complete and accurate picture of the actual state of the company. As an example, the implementation of a particular strategy (e.g., having good anti-malware tools) does not mean that the strategy is indeed effective, since its effectiveness is influenced by many other factors.

In other words, if the stakeholders' answers are evaluated with a more objective process, this will result in a more objective profiling of the company. This is our aim, which we pursue by introducing the use of some data-driven key risk indicators (KRIs) in the HTMA approach. The model we use is described next, while its application to some practical case studies is reported in Section 4.

3.1. Data-Driven KRIs. Suppose that we can dispose of a tool that monitors the company and returns a sufficient amount of quantitative evidences such as

(1) Malicious code/software activity (i.e., malware, ransomware, botnet evidences)

(2) Insecure/unencrypted/vulnerable protocols usage (i.e., P2P, vulnerable SSL, etc.)

(3) Deep web exposure (company targeted by criminals)

(4) Data breaches due to human errors, third parties, or hacking activity

(5) Software/infrastructure vulnerabilities

Clearly, we could use such information to establish a well-defined set of KRIs that negatively influence the effectiveness of the existing controls. For instance, the detection of malware incidents originating within the network of the organization is a clear indicator that some of the employees are not aware about phishing attacks and that the anti-malware tools used by the company are not enough, even if all due controls are implemented. In the same way, vulnerabilities or unnecessary/insecure services exposed on the Internet perimeter of the company are a clear indicator that the implemented controls are not effective. In both cases, it is clear that the level of security is not as high as claimed. So, the answers of the survey should be mapped

with a list of evidences pertaining to cyber incidents and technical vulnerabilities obtained, for example, with a set of cyber intelligence sources, in order to associate an effectiveness score to each one of the given answers. This way, one can get a more realistic picture of the company cyber profile: in the case of a "bad" score, the controls implemented by the company and declared by the answers are not effective. In other words, the likelihood of the associated threats should be increased, since the probability of incurring in some related incidents is higher than expected. On the contrary, in the case of a "good" score, the stakeholders' answers can be positively weighted consequently.

More precisely, when considering the stakeholders' answers, we propose to use a mapping between all the questions and the set of considered threats: this way, we can obtain a *coverage score* that in the end is used to calibrate the initial likelihood values according to the company profile. For this purpose, we first define the *control coverage vector* $\widetilde{t} = [\widetilde{t}_1, \ldots, \widetilde{t}_n]$ as the ensemble of scores computed on the basis of the stakeholders' answers. Basically, each entry $\widetilde{t}_i$ is defined as the state of coverage against the i-th considered threat. In particular, we define a *control* as a series of atomic actions to protect the organization against internal and external threats. An example of control is "Continuous Vulnerability Management," that, as per CIS v7, could be realized carrying out operations like performing authenticated vulnerability scanning, deploying automated operating system patch management tools, etc. The more actions are performed, the higher the level of control's implementation will be. For instance, an effective measurement of this control can be achieved by taking into consideration the exposure to software and infrastructure vulnerabilities; moreover, an effective measurement of the malware defenses (CSC 8) can be achieved verifying the infections events originated within the network. Then, the stakeholders' answers can be mapped into the set of controls; in particular, the answers can be collected in a vector $\mathbf{g}$, while the controls in a vector $\mathbf{d}$. Then, let $\mathbf{W}$ be a matrix with number of rows and columns equal to, respectively, the number of answers and controls. We can write

$$\mathbf{d} = \mathbf{gW}, \tag{5}$$

where the entries of $\mathbf{W}$ are all ≤ 1. With this choice, the values in $\mathbf{d}$ correspond to weighted sums of elements of $\mathbf{g}$. In particular, the entry in position (i, j) in matrix $\mathbf{W}$, which we denote as $w_{i,j}$, expresses the impact of the i-th answer on the j-th control. These coefficients need to be normalized, in the sense that the sum of each column in $\mathbf{W}$ equals 1. If $w_{i,j} = 0$, this means that the action referred to the i-th answer has no impact on the j-th control. The entries of $\mathbf{W}$ are determined on the basis of statistical analyses and evaluations on empirical data. We point out that the whole procedure is independent of both the number of collected answers and the number of analyzed controls. Indeed, choosing a different set of answers or different controls (or both of these possibilities) will just result in different dimensions and different entries for the matrix $\mathbf{W}$ (and, obviously, different lengths for the vectors $\mathbf{g}$ and $\mathbf{d}$), while the whole procedure will remain unchanged.

For each threat E_i, we then combine the elements of $\mathbf{d}$, in order to obtain the elements of $\widetilde{t}$. Indeed, each control is associated to one or more threats: the formula we have used in our simulations is

$$\widetilde{t} = \mathbf{dK}, \tag{6}$$

where $\mathbf{K}$ is another matrix in which the entries of each column sum to 1, whose entries, analogously to those of $\mathbf{W}$, are evaluated after a statistical analysis performed on empirical data. The entry in position (i, j) in $\mathbf{K}$, which we denote as $k_{i,j}$, expresses the impact of control i on the j-th threat.

By combining (5) and (6), we obtain

$$\widetilde{t} = \mathbf{gWK}, \tag{7}$$

which shows how the entries of the control coverage vector are in linear dependence with the entries of the answer vector. It must be noticed that this matrix approach is rather classic and widely used in the literature [17, 18]. The main novelty of our analysis is in the fact that the vector $\widetilde{t}$ is not used directly but it is combined with another vector resulting from empirical data.

Actually, analogously to what was done for the control coverage vector, a set of measurements can be provided in order to obtain another length-n vector that we call *effectiveness vector* and denoted as $\widehat{t} = [\widehat{t}_1, \ldots, \widehat{t}_n]$. The vectors $\widetilde{t}$ and $\widehat{t}$ are then combined, in order to obtain the *effective coverage score*, which is a length-n vector $\mathbf{t}$ whose i-th entry is obtained as $t_i = \widetilde{t}(1 - (1/5)\widehat{t}_i)_i$. It is clear that this operation corresponds to scaling the entries of the control coverage vector on the basis of the evidences found. Finally, the customized likelihoods are obtained as $p_i' = (x p_i)^{2/3}$, where $x = \max\{1 - t_i, 0.06\}$. We point out that these expressions have been derived after evaluation of available empirical data and extrapolations of the values from [19].

On the other hand, from a practical point of view, it is extremely useful to define a global indicator that measures the general cyber security posture of a company. We call this value *Security Control Score* (SCS) and we compute it as the average value of the entries of $\mathbf{t}$. Basically, a high SCS indicates substantial and effective investments in people and technology to protect the digital assets and a low exposure to costs following from cyber threats.

4. Application to Real Case Scenarios

In this section, we describe the application of our model to some real case scenarios. Consistently with the existing literature, we first consider historical data to obtain reliable values for the likelihood and impact values of a set of cyber threats. Then, we show a real case study, by applying the procedure described in Section 3, that is, introducing the evidences resulting from the intelligence tool into the estimate.

4.1. Obtaining Initial Likelihood Values. In order to provide all the required inputs for the HTMA model, we need to define a set of cyber threats for which the likelihood and

impact values can be reliably estimated. For this purpose, we rely on data reported in [19]; this choice is motivated by the fact that this report contains a large amount of information about events in 2017, so it offers a significant and recent picture of the current cyber crime panorama. According to [19], we consider nine different threats and the corresponding likelihood values; such values are listed in Table 1. An ID number is assigned to each threat, in order to simplify the notation.

For each one of the considered threats, starting from [19], we have determined the 90% confidence ranges for the impacts, on the basis of the sector in which a company operates; such ranges are listed in Table 2. For instance, for a company operating in the industrial sector, the range of losses due to malware events goes from a minimum of 1.95 M$ to a maximum of 2.19 M$.

The distinction between operating sectors might not be enough to obtain reliable estimates of the economic losses: as [19] clearly shows, the company size is another aspect that must be taken into account. In particular, coherently with [19], we can define the company size as a function of the number of seats (i.e., number of employees). This dependence can be heuristically modelled through a coefficient α defined as follows:

$$\alpha = \begin{cases} \dfrac{as^3 + bs^2 + cs}{11.7}, & \text{if } s \leq 40000, \\ \dfrac{de^{ks} + le^{ms}}{10^6}, & \text{if } s > 40000, \end{cases} \tag{8}$$

where s is the number of seats and the values of the coefficients in (8) have been obtained on the basis of extrapolations from data contained in the report ($a = 1.04 \cdot 10^{-12}$, $b = -6.54 \cdot 10^{-8}$, $c = 1.41 \cdot 10^{-3}$, $d = 1.815 \cdot 10^{7}$, $k = 2.125 \cdot 10^{-7}$, $l = 0.5838$, and $m = 6.398 \cdot 10^{-5}$).

The coefficient α can then be used to adjust the ranges listed in Table 2. In order to clarify this aspect, let us suppose that a company operating in the industrial sector has $\alpha = 0.1$: then, all the impacts reported in Table 2 must be multiplied by a coefficient equal to 0.1. For instance, the range for malware attacks becomes $[0.195; 0.219]$: these values correspond to $c_1^{(\min)}$ and $c_1^{(\max)}$ that are used in (3) and (4).

The values computed this way can then be used as reliable values for the likelihoods and impacts that are exploited in the HTMA methodology.

4.2. Case Studies. In order to validate our approach, we have run some simulations considering three different organizations O_1, O_2, and O_3, with the following common properties:

(i) There are 2000 workstations

(ii) Their business sector is industrial/manufacturing

(iii) Their annual revenue is about 350,000,000 $

It must be said, however, that the proposed approach is quite general and can be equally applied to different sectors. What usually changes passing from one scenario to another is obviously the numerical values of the quantities involved, while the approach and the set of formal relationships at the basis of the model remain substantially unchanged.

Table 1: Considered threats with the corresponding likelihood values.

ID	Threat	Likelihood
1	Malware	0.98
2	Web-based attacks	0.67
3	Denial of services	0.53
4	Malicious insiders	0.40
5	Phishing and social eng.	0.69
6	Malicious code	0.58
7	Stolen devices	0.43
8	Ransomware	0.27
9	Botnets	0.63

An intelligence tool is supposed to be used to detect cyber evidences on the cyber perimeters of O_1, O_2, and O_3, to verify the effectiveness of the security controls implemented by them. Table 3 reports the number of evidences, of the type listed in Section 3.1, monitored for the three organizations in a precise period of time (e.g., 1 year), distinguishing them on the basis of the impact: trivial, middle, and critical. Eight cyber intelligence sources S_i, with $i = 1, \ldots, 8$, have been adopted. Suitably processed, the values in Table 3 permit to determine the effectiveness vector $\widehat{t}$. By combining it with $\widetilde{t}$, we can compute the effective coverage score, $\mathbf{t}$, and finally, the security control score, SCS. All these values are reported in Table 4. Finally, the corresponding ranges of the economic losses (90% confidence intervals) are listed in Table 5.

This is all we need to run numerical simulations, following the HTMA approach, for the three considered companies. The resulting loss of exceedance curves is shown in Figure 1. These curves have been obtained by applying the theoretical approach discussed in Section 2.2. So, according to (2), each curve represents, for the specific organization it refers to, the probability that the economic loss is equal to or greater than the values of a reported in abscissa.

As it clearly results from the figure, this company profiling might have some serious consequences in the cyber risk assessment. For instance, suppose that the maximum loss that the three companies can sustain is equal to 1 M$. We see that, for the three companies, the probabilities of exceeding this value are significantly different and go from a minimum of approximately 0.05 for the company O_3 to maximum of approximately 0.75 for the company O_1. These values might be compliant or not with the profile and expectations of the organization, and in the latter case, they should suggest the adoption of correcting actions to reduce the risk. On the other hand, the picture so obtained is provisional, since it is expected to change, getting better or worse in subsequent assessment campaigns.

5. Conclusion

In this paper, we have described a data-driven approach to assess cyber risk and associate a score to the cyber exposure of a company. Our model relies on the well-known HTMA

Table 2: Impact ranges of the considered threats, expressed in M$, for different industrial sectors.

ID	Financial	Utilities and energy	Aerospace and defense	Technology and software	Health care
1	[3.51; 3.93]	[3.29; 3.69]	[2.78; 3.11]	[2.51; 2.81]	[2.4; 2.69]
2	[2.99; 3.35]	[2.8; 3.14]	[2.36; 2.65]	[2.14; 2.4]	[2.04; 2.29]
3	[2.32; 2.61]	[2.18; 2.44]	[1.84; 2.06]	[1.66; 1.86]	[1.59; 1.78]
4	[2.1; 2.35]	[1.97; 2.21]	[1.66; 1.86]	[1.5; 1.68]	[1.44; 1.61]
5	[1.93; 2.16]	[1.81; 2.03]	[1.52; 1.71]	[1.38; 1.55]	[1.32; 1.48]
6	[1.91; 2.13]	[1.79; 2]	[1.51; 1.69]	[1.37; 1.53]	[1.31; 1.46]
7	[1.28; 1.44]	[1.2; 1.35]	[1.01; 1.14]	[0.915; 1.03]	[0.875; 0.983]
8	[0.79; 0.885]	[0.74; 0.829]	[0.625; 0.7]	[0.565; 0.633]	[0.54; 0.605]
9	[0.521; 0.585]	[0.488; 0.548]	[0.413; 0.463]	[0.373; 0.418]	[0.356; 0.4]
ID	Services	Industrial	Retail	Public sector	Transportation
1	[2.11; 2.37]	[1.95; 2.19]	[1.78; 1.99]	[1.58; 1.77]	[1.4; 1.57]
2	[1.8; 2.01]	[1.66; 1.87]	[1.51; 1.7]	[1.34; 1.51]	[1.19; 1.34]
3	[1.4; 1.57]	[1.29; 1.45]	[1.18; 1.32]	[1.04; 1.17]	[0.926; 1.04]
4	[1.26; 1.42]	[1.17; 1.31]	[1.06; 1.19]	[0.944; 1.06]	[0.838; 0.939]
5	[1.16; 1.3]	[1.07; 1.21]	[0.976; 1.1]	[0.866; 0.973]	[0.769; 0.863]
6	[1.15; 1.28]	[1.06; 1.19]	[0.968; 1.08]	[0.859; 0.959]	[0.762; 0.851]
7	[0.769; 0.864]	[0.713; 0.801]	[0.648; 0.728]	[0.575; 0.646]	[0.51; 0.573]
8	[0.475; 0.532]	[0.44; 0.493]	[0.4; 0.448]	[0.355; 0.398]	[0.315; 0.353]
9	[0.314; 0.351]	[0.29; 0.326]	[0.264; 0.296]	[0.234; 0.263]	[0.208; 0.233]
ID	Consumer products	Communications	Life science	Education	Hospitality
1	[1.4; 1.57]	[1.35; 1.52]	[1.24; 1.39]	[0.955; 1.07]	[0.955; 1.07]
2	[1.19; 1.34]	[1.15; 1.29]	[1.06; 1.19]	[0.813; 0.912]	[0.813; 0.912]
3	[0.926; 1.04]	[0.897; 1.01]	[0.823; 0.924]	[0.632; 0.709]	[0.632; 0.709]
4	[0.838; 0.939]	[0.811; 0.909]	[0.745; 0.834]	[0.572; 0.641]	[0.572; 0.641]
5	[0.769; 0.863]	[0.744; 0.836]	[0.683; 0.767]	[0.525; 0.589]	[0.525; 0.589]
6	[0.762; 0.851]	[0.738; 0.824]	[0.678; 0.756]	[0.52; 0.581]	[0.52; 0.581]
7	[0.51; 0.573]	[0.494; 0.555]	[0.454; 0.51]	[0.348; 0.391]	[0.348; 0.391]
8	[0.315; 0.353]	[0.305; 0.342]	[0.28; 0.314]	[0.215; 0.241]	[0.215; 0.241]
9	[0.208; 0.233]	[0.201; 0.226]	[0.185; 0.207]	[0.142; 0.159]	[0.142; 0.159]

Table 3: Evidences of intelligence.

Source	Evidences								
	Trivial			Middle			Critical		
	O_1	O_2	O_3	O_1	O_2	O_3	O_1	O_2	O_3
S_1	12	5	1	12	5	1	12	5	1
S_2	13	1	2	12	5	1	12	5	1
S_3	21	2	0	12	5	1	12	5	1
S_4	3	2	0	12	5	1	12	5	1
S_5	14	9	2	12	5	1	12	5	1
S_6	5	1	0	12	5	1	12	5	1
S_7	23	0	1	12	5	1	12	5	1
S_8	7	1	3	12	5	1	12	5	1

Table 4: Effective coverage score values for the three simulated companies.

Threat ID	O_1	O_2	O_3
1	0.5	0.61	0.85
2	0.23	0.79	0.78
3	0.64	0.74	0.95
4	0.12	0.34	0.67
5	0.21	0.87	0.90
6	0.16	0.56	0.75
7	0.15	0.23	0.57
8	0.68	0.75	0.91
9	0.91	0.87	0.82
SCS	0.4	0.64	0.8

Table 5: Ranges of economic losses for each considered organization (k$).

Threat ID	O_1 (0.4)	O_2 (0.64)	O_3 (0.8)
1	[397.53; 446.46]	[417.31; 468.68]	[209.13; 234.86]
2	[355.51; 400.48]	[175.87; 198.12]	[171.17; 192.82]
3	[136.53; 153.46]	[138.41; 155.58]	[49.15; 55.24]
4	[189.65; 212.34]	[192.48; 215.51]	[110.39; 123.60]
5	[229.01; 258.98]	[84.09; 95.10]	[67.10; 75.89]
6	[217.65; 244.34]	[171.48; 192.51]	[106.47; 119.52]
7	[120.56; 135.43]	[139.39; 156.60]	[87.31; 98.08]
8	[28.01; 31.38]	[30.55; 34.24]	[13.86; 15.53]
9	[13.93; 15.66]	[22.31; 25.0851]	[24.95; 28.04]

approach but reduces the margin for subjectivity by introducing the use of some quantitative key risk indicators. Its applicability in practice has been illustrated through some preliminary case studies taken from the industrial manufacturing world. Future works will concern application of the proposed method in a variety of different sectors, with the aim to catch practical evidences of the advantages it offers with respect to previous methods.

The proposed model should allow overcoming the intrinsic limits of the existing risk assessment approaches, which are based on the estimate of the threats' occurrence probability and on the observation of events that occurred in the past, thus not guaranteeing an adequate protection for the future. The proposed approach is also expected to

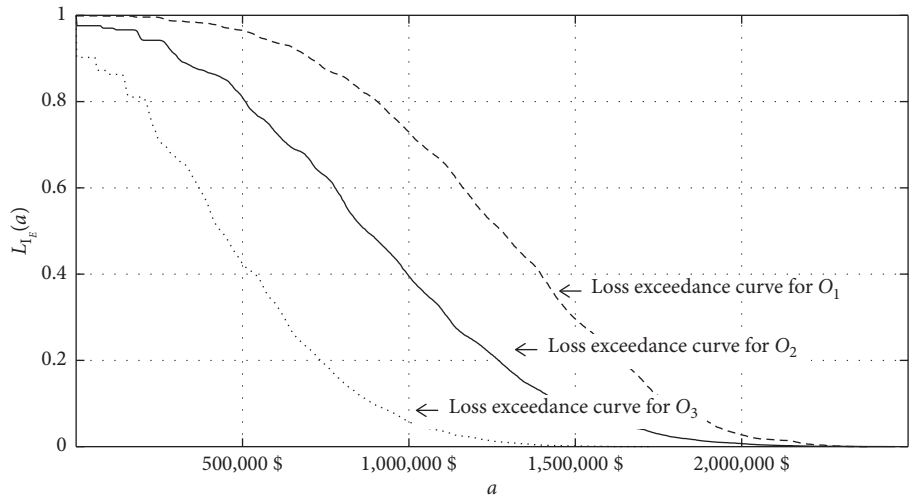


Figure 1: Loss exceedance curves for the three considered organizations.

provide companies and institutions with a new practical tool able to assess their cyber risk exposure and helping the definition of data protection policies for processes, systems, and infrastructures. The proposed solution is wide-ranging and applicable to different contexts, maintaining versatility and possibility to be used by companies and institutions of different size and working in different fields.

Data Availability

All data used in the paper come from cited references or are reported in the paper.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. Ugur Aksu, M. Hadi Dilek, E. Islam Tatli et al., "A quantitative CVSS-based cyber security risk assessment methodology for IT systems," in *Proceedings of the 2017 International Carnahan Conference on Security Technology*, pp. 1–8, ICCST, Madrid, Spain, October 2017.

[2] K. Hartmann and C. Steup, "The vulnerability of UAVs to cyber attacks—an approach to the risk assessment," in *Proceedings of the 5th International Conference on Cyber Conflict*, pp. 1–23, CYCON, Tallinn, Estonia, June 2013.

[3] B. Karabacak and I. Sogukpinar, "ISRAM: information security risk analysis method," *Computers & Security*, vol. 24, no. 2, pp. 147–159, 2005.

[4] S. Naumov and I. Kabanov, "Dynamic framework for assessing cyber security risks in a changing environment," in *Proceedings of the 2016 International Conference on Information Science and Communications Technologies (ICISCT)*, pp. 1–4, Tashkent, Uzbekistan, November 2016.

[5] C. Biener, M. Eling, and J. H. Wirfs, "Insurability of cyber risk: an empirical analysis," *The Geneva Papers on Risk and Insurance—Issues and Practice*, vol. 40, no. 1, pp. 131–158, 2015.

[6] A. Rot, "IT risk assessment: quantitative and qualitative approach," in *Proceedings of the World Congress on Engineering and Computer Science 2008 (WCECS 2008)*, San Francisco, CA, USA, October 2008.

[7] R. Bojanc and B. Jerman-Blažič, "An economic modelling approach to information security risk management," *International Journal of Information Management*, vol. 28, no. 5, pp. 413–422, 2008.

[8] S. Romanosky, "Examining the costs and causes of cyber incidents," *Journal of Cybersecurity*, vol. 2, no. 2, pp. 121–135, 2016.

[9] D. W. Hubbard and R. Seiersen, *How to Measure Anything in Cybersecurity Risk*, Wiley, Hoboken, NJ, USA, 2016.

[10] J. Freund and J. Jones, *Measuring and Managing Information Risk—A FAIR Approach*, Butterworth-Heinemann, Oxford, UK, 2015.

[11] M. McNeil, T. Llansó, and D. Pearson, "Application of capability-based cyber risk assessment methodology to a space system," in *Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security (HoTSoS'18)*, pp. 1–10, ACM, Raleigh, NC, USA, April 2018.

[12] M.-E. Paté-Cornell, M. Kuypers, M. Smith, and P. Keller, "Cyber risk management for critical infrastructure: a risk analysis model and three case studies," *Risk Analysis*, vol. 38, no. 2, pp. 226–241, 2018.

[13] M. Evans, L. A. Maglaras, Y. He, and H. Janicke, "Human behaviour as an aspect of cybersecurity assurance," *Security and Communication Networks*, vol. 9, no. 17, pp. 4667–4679, 2016.

[14] M. Eling and J. Wirfs, "What are the actual costs of cyber risk events?," *European Journal of Operational Research*, vol. 272, no. 3, pp. 1109–1119, 2019.

[15] B. Edwards, S. Hofmeyr, and S. Forrest, "Hype and heavy tails: a closer look at data breaches," *Journal of Cybersecurity*, vol. 2, no. 1, pp. 3–14, 2016.

[16] A. Baze, "Realistic risk management using the CIS 20 security controls," Technical Report, SANS Institute, Bethesda, MA, USA, 2016.

[17] S. Baddelpeli and G. Vert, "Adaptive security metrics for computer systems," in *Proceedings of the 2006 International Conference on Security & Management*, pp. 351–356, SAM, Las Vegas, NV, USA, June 2006.

[18] M. Jouini and L. B. A. Rabai, "Mean failure cost extension model towards security threats assessment: a cloud computing

case study," *Journal of Computers*, vol. 10, no. 3, pp. 184–194, 2015.

[19] Ponemon Institute LLC—Accenture, *Cost of Cyber Crime Study*, Technical Report, Ponemon Institute LLC—Accenture, North Traverse City, MI, USA, 2017.